UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

ALGORITHMES POUR LE PROBLÈME D'AFFECTATION DU TRAFIC D'ÉQUILIBRE AVEC DEMANDES FIXES COMPARAISONS

par

Jacques Guélat

Université de Montréal
Centre de recherche sur les transports - Publication #299
Mai 1983

Dépôt légal - 4e trimestre 1983

RÉSUMÉ

Le présent rapport contient la formulation et les résultats comparatifs de l'application de cinq algorithmes de résolution du problème d'affectation du trafic d'équilibre avec demandes fixes.

- - - - - - - - - - - - - - -

TABLE DES MATIÈRES

1. Introduction 1

2. Le problème. 2

3. Les algorithmes de résolution du problème. 5
   a) Algorithme de Frank & Wolfe 5
   b) Méthode heuristique 11
   c) La technique PARTAN 11
   d) Stratégie de restriction. 26
   e) Approximation quadratique 33

4. Réseaux tests. 41

5. Résultats et commentaires. 43
   - Réseau Dupuis. 44
   - Réseau de Hull 49
   - Réseau de Winnipeg 67

6. Conclusions. 74

Références 75

Annexe A 77

Annexe B 82

- - - - - - - - - - - - - -

1. INTRODUCTION

Le but du présent rapport est de comparer l'efficacité de plusieurs variantes de l'algorithme de Frank & Wolfe appliquées au problème d'affectation du trafic d'équilibre avec demandes fixes. Un bref exposé de ce problème classique est fourni au début de ce rapport. Le choix s'est porté sur les quatre variantes suivantes:

a) heuristique : issue d'une idée de A. Weintraub, cette méthode consiste en une légère modification de l'algorithme de Frank & Wolfe (dénoté F & W dans la suite);

b) technique PARTAN : génération d'une nouvelle direction de recherche à l'aide des deux derniers points trouvés. On fait alors alterner les directions F & W et PARTAN;

c) stratégie de restriction : elle consiste à minimiser la fonction économique sur un domaine restreint avant de poursuivre un algorithme standard;

d) approximation quadratique : génération des directions de descente à l'aide de l'approximation quadratique de la fonction économique (et non de l'approximation linéaire comme dans F & W). Cette idée est due à R.S. Dembo [4].

Après la formulation de ces variantes, on trouvera dans ce rapport les résultats des tests effectués sur des réseaux de différentes tailles. Les programmes ont été écrits en FORTRAN 4 et consistent en des adaptations du programme "TRAFFIC" (cf. Nguyen et James [15]). Le code est donné en annexe B. L'ordinateur utilisé est le CDC Cyber 835 de l'Université de Montréal.

Ce travail représente une version corrigée et augmentée du rapport technique "Algorithmes pour le problème d'affection du trafic avec demandes fixes - comparaisons" fourni en octobre 1982.

2. LE PROBLÈME

Le problème considéré ici est celui de la recherche d'un flot d'équilibre dans un réseau à un seul mode, à fonctions de coûts séparables par arc et à demandes fixes.

Le principe d'équilibre utilisé, énoncé par Wardrop [20], est celui de type descriptif (comportement égoïste de l'usager) : "les coûts des chemins utilisés sont égaux et ceux des chemins non-utilisés leur sont supérieurs ou égaux".

Formulation du problème

N : ensemble des noeuds (sommets) du réseau; $|N| = n$

A : ensemble des arcs (liens) du réseau; $|A| = m$

$v_{m \times 1} = (v_a)$: vecteur des flots sur les liens $a \in A$

$s_a(v_a)$: fonction de coût associée à l'arc $a \in A$

I : ensemble des paires O/D

K_i : ensemble des chemins reliant l'origine à la destination de la i-ième paire O/D

h_k : flot sur le chemin k

$\bar{g}_i$: demande (fixe) associée à la i-ième paire O/D.

On a évidemment la relation :

$$v_a = \sum_{i \in I} \sum_{k \in K_i} \delta_{ak} \, h_k \qquad (1)$$

où $$\delta_{ak} = \begin{cases} 1 & \text{si } a \in k \\ 0 & \text{sinon} \end{cases}$$

On supposera les fonctions de coût strictement monotones croissantes, ce qui assurera l'unicité de la solution (cf. Aashtiani et Magnanti [1], Smith [18]).

En posant

$$s_k = \text{coût du chemin } k = \sum_{a \in A} \delta_{ak} \, s_a(v_a)$$

et $$u_i = \min_{k \in K_i} s_k,$$

le principe d'équilibre de Wardrop s'énonce ainsi :

$$s_k^* - u_i^* \begin{cases} = 0 & \text{si } h_k^* > 0 \\ \geq 0 & \text{si } h_k^* = 0 \end{cases} \qquad k \in K_i, \; i \in I \qquad (2)$$

où h_k^* est, en particulier, un flot réalisable, c'est-à-dire satisfaisant à :

conservation de flot : $$\sum_{k \in K_i} h_k = \bar{g}_i, \; i \in I \qquad (3)$$

non-négativité : $$h_k \geq 0, \quad k \in K_i, \; i \in I \qquad (4)$$

(3) et (4) définissent le domaine réalisable Ω qu'on supposera non-vide.

On montre (cf. Florian [6]) que la formulation (2) du principe d'équilibre possède les deux formes équivalentes suivantes (inégalités variationnelles) :

chemins : $(s_k^*-u_i^*)(h_k-h_k^*) \geq 0, \quad k\in K_i,\ i\in I,\ h_k\in\Omega$ (5)

arcs : $\sum_{a\in A} s_a(v_a^*)(v_a-v_a^*) \geq 0,\ v_a$ réalisable ((1),(3),(4)) (6)

Comme on ne considère ici que des coûts séparables par arc (i.e. $S_a(v) = S_a(v_a)$), le jacobien Jac (S(v)) est diagonal, donc symétrique. Il existe alors un problème de minimisation équivalent à la forme variationnelle (6) :

$$\left.\begin{array}{l} \text{Min } F = \sum_{a\in A} \int_0^{v_a} s_a(v)dv \\ \text{sà : } (1)(3)(4) \end{array}\right\} \quad (7)$$

C'est sous cette forme que nous allons résoudre le problème.

Remarques :

1) Les fonctions de coût étant strictement monotones, la fonction objectif F est convexe. (7) est donc un problème de minimisation d'une fonction convexe sujette à des contraintes linéaires. Les méthodes de résolution analysées plus loin s'appliquent de façon générale à tous les problèmes de ce type.

2) On peut facilement généraliser la formulation du problème d'équilibre du trafic au cas des demandes variables (cf. Florian [6,7], Nguyen [14]), à plusieurs commodités (cf. Dafermos [3]), ou encore avec coûts asymétriques (cf. Smith [18], Florian [7]), la résolution du problème dans ce dernier cas étant rendue plus ardue car l'existence d'un problème équivalent de minimisation n'est plus assurée (cf. Nguyen et Dupuis [16] pour un autre mode de résolution).

3. LES ALGORITHMES DE RÉSOLUTION DU PROBLÈME

a) Algorithme de Frank & Wolfe [8]

Soit $F : E^n \to E^1$ différentiable et convexe. Considérons le problème de minimisation à contraintes linéaires :

$$\left.\begin{array}{l} \text{Min } F(x) \\ \text{sà : } Ax = b \\ \qquad x \geq 0 \end{array}\right\} \qquad (8)$$

Soit x^ℓ la solution réalisable courante. On linéarise alors le problème pour chercher une direction réalisable de descente en x^ℓ. (Par convention, lorsqu'on parlera de direction de descente, on supposera qu'elle est réalisable.) On résoud alors le problème (8) restreint à la direction trouvée, ce qui fournit le point $x^{\ell+1}$.

L'algorithme prend donc la forme :

0. trouver une solution réalisable initiale x^1; $\ell := 1$;
1. recherche d'une direction de descente d^ℓ par le biais de l'approximation linéaire de F en x^ℓ;
2. recherche unidimensionnelle du min. de F dans la direction d^ℓ tout en restant dans le domaine réalisable, ce qui fournit $x^{\ell+1}$.

Plus formellement et en introduisant un critère d'arrêt, cet algorithme s'écrit :

0. (*initialisation*)
   trouver une solution réalisable x^1; $\ell := 1$;

1. (*direction de descente*)

résoudre le problème linéaire

$$\left.\begin{array}{ll} \text{Min } F(x^\ell)+\nabla F(x^\ell)(y-x^\ell) & \\ \text{sà :} \quad Ay = b & \\ \qquad\quad y \geq 0 & \end{array}\right\} \iff \left.\begin{array}{l} \text{Min } \nabla F(x^\ell)y \\ \text{sà : } Ay = b \\ \qquad\quad y \geq 0 \end{array}\right\}$$

soit y^ℓ la (une) solution optimale (y^ℓ est un point extrême)

critère d'arrêt : si $\left|\dfrac{\nabla F(x^\ell)(y^\ell-x^\ell)}{F(x^\ell)}\right| < \varepsilon$ alors STOP

$d^\ell := y^\ell - x^\ell$

2. (*recherche unidimensionnelle*)

résoudre le problème

$$\min_{0\leq\lambda\leq 1} F(x^\ell+\lambda d^\ell)$$

soit λ_ℓ la solution obtenue; λ_ℓ = "pas F & W"

$x^{\ell+1} := x^\ell + \lambda_\ell d^\ell$

$\ell := \ell+1$; retour à 1.

La direction d^ℓ trouvée à l'étape 1 est une direction de descente (cf. Nguyen [14]). Cet algorithme converge de façon linéaire vers un point de Kuhn-Tucker (cf. Frank & Wolfe [8], Bazaraa et Shetty [2]).

Remarque concernant le critère d'arrêt :

F étant convexe, on a :

$F(x^*) > F(x^\ell) + \nabla F(x^\ell)(x^*-x^\ell)$ où x^* sol. opt. de (8)

Mais, par construction de y^ℓ, on a :

$\nabla F(x^\ell)(x^*-x^\ell) \geq \nabla F(x^\ell)(y^\ell-x^\ell)$

d'où : $F(x^*) \geq F(x^\ell) + \nabla F(x^\ell)(y^\ell-x^\ell)$ (9)

Le terme de droite fournit, à chaque itération de l'algorithme, une borne inférieure pour l'optimum de F. On conservera la meilleure des bornes inférieures ainsi générées.

(9) s'écrit : $\underbrace{F(x^\ell)-F(x^*)}_{\geq 0} \leq -\nabla F(x^\ell)(y^\ell-x^\ell)$.

On obtient donc une majoration de l'accroissement absolu de F à l'étape ℓ :

$$|F(x^\ell)-F(x^*)| \leq |\nabla F(x^\ell)(y^\ell-x^\ell)| .$$

Divisons par $|F(x^\ell)|$ pour avoir l'accroissement relatif :

$$\left|\frac{F(x^\ell)-F(x^*)}{F(x^\ell)}\right| \leq \left|\frac{\nabla F(x^\ell)(y^\ell-x^\ell)}{F(x^\ell)}\right| . \qquad (10)$$

Cette inégalité fournit le critère d'arrêt utilisé dans l'algorithme.

Finalement, traduisons cet algorithme dans le cadre du problème d'affectation du trafic :

$$F(v) = \sum_{a\in A} \int_0^{v_a} s_a(v)dv = \sum_{a\in A} S_a(v_a)$$

$$\nabla F(v) = (s_a(v_a)) .$$

Le sous-problème linéaire de l'étape 1 de l'algorithme est un problème de plus courts chemins. En effet ce sous-problème s'écrit :

$$\left.\begin{array}{l} \text{Min} \quad \sum_{a\in A} s_a(v_a^\ell)\, y_a \\ \text{sà} : \sum_{k\in K_i} h_k = \bar{g}_i, \quad i\in I \\ \qquad\quad h_k \geq 0, \quad k\in K_i,\ i\in I \\ y_a = \sum_{i\in I} \sum_{k\in K_i} \delta_{ak}\, h_k \end{array}\right\} \qquad (11)$$

$$\text{or :} \quad \sum_{a\in A} s_a(v_a^\ell) y_a = \sum_{a\in A} s_a(v_a^\ell) \sum_{i\in I} \sum_{k\in K_i} \delta_{ak}\, h_k$$

$$= \sum_{i\in I} \sum_{k\in K_i} h_k \sum_{a\in A} \delta_{ak}\, s_a(v_a^\ell)$$

$$= \sum_{i\in I} \sum_{k\in K_i} h_k\, s_k^\ell$$

où : $s_k = \sum_{a\in A} \delta_{ak} s_a(v_a^\ell)$ = coût du chemin k

(11) est donc équivalent à :

$$\left.\begin{array}{l} \text{Min} \quad \sum_{i\in I} \sum_{k\in K_i} s_k^\ell\, h_k \\ \text{sà} : \sum_{k\in K_i} h_k = \bar{g}_i, \quad i\in I \\ \qquad\quad h_k \geq 0, \quad k\in K_i,\ i\in I \end{array}\right\} \qquad (12)$$

(12) est un problème de flot à coût minimal sans capacité sur les arcs. On obtient donc une solution optimale en faisant passer tout le flot sur le chemin le plus court (au sens des coûts).

Soit h_k^ℓ cette solution "tout ou rien", c'est-à-dire

$$h_k^\ell = \begin{cases} \bar{g}_i & \text{si } k \text{ est le plus court chemin de } K_i \\ 0 & \text{sinon.} \end{cases}$$

Alors $y^\ell = (y_a^\ell) = \left(\sum_{i \in I} \sum_{k \in K_i} \delta_{ak} \, h_k^\ell \right)$.

L'utilisation d'algorithmes du type F & W pour la résolution du problème d'équilibre du trafic se justifie par la forme très simple du sous-problème linéarisé et par l'existence d'algorithmes efficaces pour le résoudre.

En référence avec le programme utilisé dans les applications (cf. annexe B), appelons EXFLOW la routine qui calcule le flot extrémal y^ℓ à partir des coûts $s_a(v_a^\ell)$.

$$y^\ell = \text{EXFLOW} \, (s_a(v_a^\ell)) \ .$$

Comme solution initiale à l'étape 0 de l'algorithme, nous prendrons le flot (extrémal) obtenu par EXFLOW à partir des coûts $s_a(0)$. Il est évident que n'importe quelle autre solution réalisable convient.

L'algorithme prend donc la forme :

<u>Algorithme de Frank & Wolfe pour le problème d'affectation du trafic d'équilibre</u>

0. (*initialisation*)

$v^1 := \text{EXFLOW} \, (s_a(0))$; $\ell := 1$;

1. (*direction de descente*)

$y^{\ell} := \text{EXFLOW}\ (s_a(v^{\ell}))$

$d^{\ell} := y^{\ell} - v^{\ell}$

$$\text{Si} \left| \frac{\sum_{a \in A} s_a(v_a^{\ell}) d_a^{\ell}}{\sum_{a \in A} S_a(v_a^{\ell})} \right| < \varepsilon \text{ , alors STOP}$$

2. (*recherche unidimensionnelle*)

λ_{ℓ} solution optimale de $\underset{0 \le \lambda \le 1}{\text{Min}} \sum_{a \in A} S_a(v_a^{\ell} + \lambda d_a^{\ell})$

$v^{\ell+1} := v^{\ell} + \lambda_{\ell} d^{\ell}$

$\ell := \ell+1$; retour à 1.

Une discussion des méthodes de résolution du problème de minimisation unidimensionnel de l'étape 2 se trouve à l'annexe A.

Remarque :

$$(9) \Rightarrow \nabla F(x^{\ell})(x^{\ell} - y^{\ell}) \ge F(x^{\ell}) - F(x^*) \ge 0 \ .$$

Donc $\Delta^{\ell} = \nabla F(x^{\ell})x^{\ell} - \nabla F(x^{\ell})y^{\ell}$ est une mesure de l'"optimalité" de x^{ℓ}.

Dans le cas du modèle de trafic d'équilibre, on obtient :

$$\nabla F(v^{\ell})v^{\ell} = \sum_{a \in A} s_a(v_a^{\ell}) v_a^{\ell}$$

= coût total au point v^{ℓ}

$$\nabla F(v^{\ell})y^{\ell} = \sum_{a \in A} s_a(v_a^{\ell}) y_a^{\ell}$$

= coût total au point y^{ℓ}

$$= \sum_{a \in A} s_a(v_a^\ell) \sum_{i \in I} \sum_{k \in K_i} \delta_{ak} h_k^\ell \qquad (h_k^\ell : \text{solution "tout ou rien"})$$

$$= \sum_{i \in I} \sum_{k \in K_i} h_k^\ell s_k^\ell$$

$$= \sum_{i \in I} \bar{g}_i u_i^\ell$$

d'où : $$\Delta^\ell = \sum_{a \in A} s_a(v_a^\ell) v_a^\ell - \sum_{i \in I} \bar{g}_i u_i^\ell \qquad (13)$$

b) Méthode heuristique

Il s'agit d'une légère modification de l'algorithme de F & W régulier proposée par Weintraub. Après un certain nombre d'itérations régulières de F & W, le pas λ_ℓ devient petit. L'idée de la méthode est alors de fausser (à intervalles réguliers d'itérations) le résultat fourni par la recherche unidimensionnelle en multipliant le pas λ_ℓ par un coefficient $C > 1$, en espérant ainsi obtenir une meilleure direction de descente en $x^{\ell+1}$.

$$\lambda_\ell^{mod} := \lambda_\ell * C .$$

La décroissance de la fonction économique n'étant plus assurée, on effectuera un faux pas de F & W uniquement si $F(x^\ell + \lambda_\ell^{mod} y^\ell) \leq F(x^\ell)$.

c) La technique PARTAN

Il est connu (cf. Luenberger [13]) que près de la solution optimale la méthode de F & W converge "mal". Les solutions ont tendance à osciller :

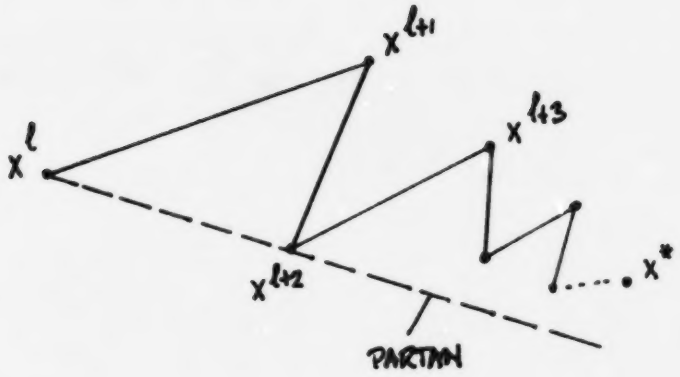

Figure 1

D'où l'idée de générer, à chaque itération de l'algorithme de F &W, une nouvelle direction de recherche : $d_p = x^{l+2} - x^l$.

L'algorithme s'énonce alors :

0. Trouver une solution initiale x^1

   générer x^2 par F &W; $l=2$.

1. Générer z^l par F &W.

2. Direction de recherche PARTAN : $d_p^l := z^l - x^{l-1}$

   une recherche unidimensionnelle fournit alors x^{l+1}

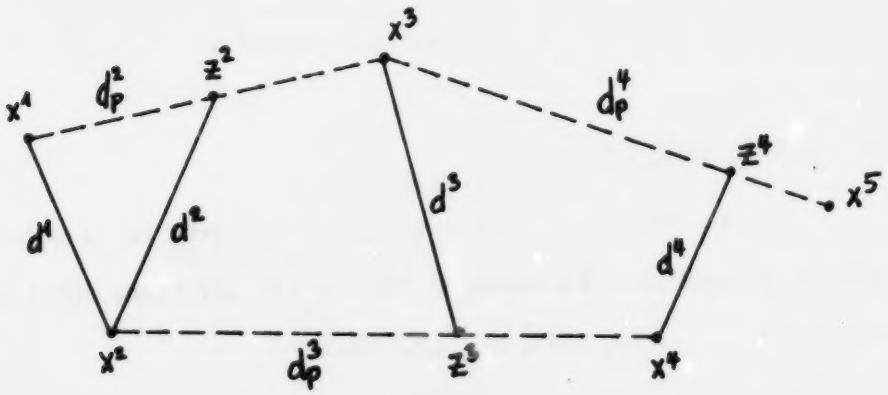

Figure 2

Cette technique, introduite par Shah-Buehler-Kempthorne [17] pour des problèmes non contraints, est, dans le cas de fonctions économiques quadratiques, du type "directions conjuguées". Ce sont les directions (x^2-x^1), (x^3-x^2), (x^4-x^3), ..., $(x^{\ell+1}-x^\ell)$ qui sont alors conjuguées. Les points intermédiaires z^ℓ sont trouvés par une méthode de "steepest descent" et plus généralement en imposant que les directions $(z^\ell-x^\ell)$ soient parallèles à $\Pi_1, \Pi_2, \ldots, \Pi_{\ell-1}$ où Π_i est l'hyperplan tangent au contour de la fonction économique en x^i. D'où le nom "parallel tangents", abrégé PARTAN, attribué à cette technique. On trouve dans Sorensen [19] une comparaison de PARTAN avec les autres techniques de directions conjuguées.

Un peu plus formellement, l'algorithme PARTAN pour la résolution du problème (8) s'écrit :

0. (*initialisation*)
   trouver une solution réalisable x^1; $\ell := 1$.

1. (*direction de descente F & W*)
   $d^\ell := y^\ell - x^\ell$; critère d'arrêt.

2. (*recherche unidimensionnelle F & W*)
   pas F & W : λ_ℓ
   $z^\ell := x^\ell + \lambda_\ell d^\ell$.

3. (*recherche dans la direction PARTAN*)
   Si $\ell = 1$ alors $x^{\ell+1} := z^\ell$
   Sinon direction de recherche PARTAN : $d_p^\ell := z^\ell - x^{\ell-1}$
   calcul du pas maximal p^{max} dans cette direction
   recherche unidimensionnelle :

résoudre $\min_{0 \le p \le p^{max}} F(x^{\ell-1} + p d_p^{\ell})$

soit p_ℓ la solution optimale

p_ℓ = "pas PARTAN"

$x^{\ell+1} := x^{\ell-1} + p_\ell d_p^{\ell}$

$\ell := \ell + 1$

retour à 1.

L'implantation de cet algorithme nécessite la mémorisation des deux solutions les plus récentes x^{ℓ} et $x^{\ell-1}$.

Calcul du pas PARTAN maximal

Les calculs qui suivent sont une généralisation de ceux fournis dans LeBlanc, Helgason et Boyce [12]. En effet, dans cet article, on ne donne que les cinq premiers pas maximaux. Nous exhiberons ici une formule générale (sous forme inductive) fournissant le pas maximal à l'itération ℓ ($\ell > 1$).

Etape 2 : $z^{\ell} = x^{\ell} + \lambda_\ell (y^{\ell} - x^{\ell})$ (14)

Etape 3 : $x^{\ell+1} = x^{\ell-1} + p_\ell (z^{\ell} - x^{\ell-1})$ (15)

d'où : $x^{\ell+1} = p_\ell z^{\ell} + (1-p_\ell) x^{\ell-1}$

$= p_\ell \lambda_\ell y^{\ell} + p_\ell (1-\lambda_\ell) x^{\ell} + (1-p_\ell) x^{\ell-1}$.

Introduisons, pour alléger quelque peu la notation, la convention : $\bar{a} = 1 - a$.

Alors : $$x^{\ell+1} = p_\ell \lambda_\ell y^\ell + p_\ell \bar{\lambda}_\ell x^\ell + \bar{p}_\ell x^{\ell-1} \qquad (16)$$

Soit $\{y^1, y^2, \ldots, y^{\ell}\}$ la liste des points extrémaux générés à l'étape 1 jusqu'ici. Ajoutons-y x^1 (qui sera aussi extrémal dans les applications, ceci n'étant évidemment pas une condition nécessaire) et soit C_ℓ l'enveloppe convexe des points de cette liste :

$$C_\ell = \text{Conv } \{x^1, y^1, \ldots, y^\ell\} \quad ; \quad C_0 = \{x^1\}$$

Dénotons x^1 par y^0. $C_\ell = \text{Conv } \{y^0, \ldots, y^\ell\}$

On voit que

$$x^1 = y^0 \in C_0$$

$$x^2 \in C_1 = \text{Conv } \{y^0, y^1\}$$

$$(x^2 = x^1 + \lambda_1(y^1 - x^1) = \lambda_1 y^1 + (1-\lambda_1)x^1 \quad \text{et} \quad 0 \le \lambda_1 \le 1)$$

On va alors définir p_ℓ^{max} de telle manière que $x^{\ell+1} \in C_\ell$. Le pas ainsi défini n'est pas le plus grand possible pour que $x^{\ell+1}$ reste réalisable comme le montre l'exemple bidimensionnel suivant :

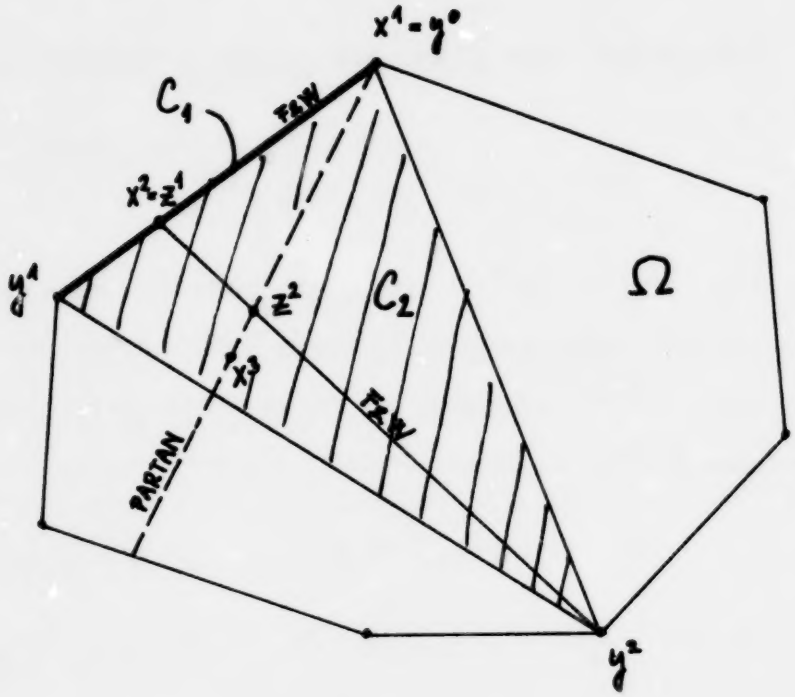

Figure 3

En général $C_\ell \subsetneq \Omega$ domaine réalisable. Comme il n'est pas possible de connaître tous les points extrémaux en pratique (en nombre prohibitif!), on se contentera donc du pas maximal dans C_ℓ.

L'équation (16) nous dit que $x^{\ell+1} \in L\ (y^0,\ldots,y^\ell)$ = enveloppe linéaire de ces points. La démonstration se fait clairement par induction en remarquant que

$$L(y^0,\ldots,y^{\ell-2}) \subset L(y^0,\ldots,y^{\ell-1}) \subset L(y^0,\ldots,y^\ell)$$

Donc $$x^{\ell+1} = \sum_{i=0}^{\ell} w_{\ell+1}^i\ y^i \qquad (17)$$

$w_{\ell+1}^i$ sont les poids des différents points extrêmes.

$$\sum_{i=0}^{\ell} w_{\ell+1}^{i} = 1 \quad , \qquad \forall\, \ell \tag{18}$$

<u>Démonstration</u> : par induction sur ℓ

$$x^1 = 1 \cdot x^1 = 1 \cdot y^0 \qquad \Rightarrow \quad w_1^0 = 1 \tag{19}$$

$$x^2 = z^1 = (1-\lambda_1)x^1 + \lambda_1 y^1 \Rightarrow \left.\begin{array}{l} w_2^0 = \bar{\lambda}_1 \\ w_2^1 = \lambda_1 \end{array}\right\} \tag{20}$$

d'où : $\quad w_2^0 + w_2^1 = 1$

(16) : $\quad x^{\ell+1} = p_\ell \lambda_\ell y^\ell + p_\ell \bar{\lambda}_\ell x^\ell + \bar{p}_\ell x^{\ell-1}$

$$= p_\ell \lambda_\ell y^\ell + p_\ell \bar{\lambda}_\ell \sum_{i=0}^{\ell-1} w_\ell^i y^i + \bar{p}_\ell \sum_{i=0}^{\ell-2} w_{\ell-1}^i y^i$$

$$= p_\ell \lambda_\ell y^\ell + p_\ell \bar{\lambda}_\ell w_\ell^{\ell-1} y^{\ell-1} + \sum_{i=0}^{\ell-2} (p_\ell \bar{\lambda}_\ell w_\ell^i + \bar{p}_\ell w_{\ell-1}^i) y^i$$

d'où :
$$\left.\begin{array}{ll} w_{\ell+1}^{\ell} = p_\ell \lambda_\ell & \\ w_{\ell+1}^{\ell-1} = p_\ell \bar{\lambda}_\ell w_\ell^{\ell-1} & \\ w_{\ell+1}^{i} = p_\ell \bar{\lambda}_\ell w_\ell^i + \bar{p}_\ell w_{\ell-1}^i \quad , & 0 \le i \le \ell-2 \end{array}\right\} \tag{21}$$

et $$\sum_{i=0}^{\ell} w_{\ell+1}^{i} = p_\ell \lambda_\ell + p_\ell \bar{\lambda}_\ell \underbrace{\sum_{i=0}^{\ell-1} w_\ell^i}_{1} + \bar{p}_\ell \underbrace{\sum_{i=0}^{\ell-2} w_{\ell-1}^i}_{1}$$

$$= p_\ell(\lambda_\ell + \bar{\lambda}_\ell) + \bar{p}_\ell = p_\ell + \bar{p}_\ell = 1 \qquad \square$$

Les équations (19)-(21) fournissent les formules récursives pour les poids des points extrêmes dans l'expression (17) de $x^{\ell+1}$.

On a donc finalement :

$$x^{\ell+1} \in C_\ell \iff w_{\ell+1}^i \geq 0 \quad , \quad 0 \leq i \leq \ell \tag{22}$$

Les conditions $w_{\ell+1}^i \geq 0$, $0 \leq i \leq \ell$, s'écrivent :

$\underline{i = \ell}$: $w_{\ell+1}^{\ell} = p_\ell \lambda_\ell \geq 0$

Cette condition est automatiquement vérifiée car $0 \leq \lambda_\ell \leq 1$ et on veut un pas p_ℓ non-négatif!

$\underline{i=\ell-1}$: $w_{\ell+1}^{\ell-1} = p_\ell \bar{\lambda}_\ell w_\ell^{\ell-1} \geq 0$

A nouveau, cette condition est automatiquement vérifiée par construction.

$\underline{0 \leq i \leq \ell-2}$: $w_{\ell+1}^i = p_\ell \bar{\lambda}_\ell w_\ell^i + \bar{p}_\ell w_{\ell-1}^i \geq 0$

ou : $p_\ell (\bar{\lambda}_\ell w_\ell^i - w_{\ell-1}^i) \geq -w_{\ell-1}^i$

Si $(\bar{\lambda}_\ell w_\ell^i - w_{\ell-1}^i) \geq 0$, comme $-w_{\ell-1}^i \leq 0$, alors l'inéquation ci-dessus admet un $p_\ell \geq 0$ aussi grand qu'on veut, c'est-à-dire pas de borne sur p_ℓ.

Si, par contre, $(\bar{\lambda}_\ell w_\ell^i - w_{\ell-1}^i) < 0$, alors l'inéquation ci-dessus s'écrit :

$$p_\ell \leq \frac{-w^i_{\ell-1}}{\bar{\lambda}_\ell w^i_\ell - w^i_{\ell-1}} = \frac{w^i_{\ell-1}}{w^i_{\ell-1} - \bar{\lambda}_\ell w^i_\ell}$$

D'où :

$$\boxed{p^{max}_\ell = \min_{0\leq i\leq \ell-2} \left\{ \frac{w^i_{\ell-1}}{w^i_{\ell-1} - \bar{\lambda}_\ell w^i_\ell} \;\middle|\; w^i_{\ell-1} - \bar{\lambda}_\ell w^i_\ell > 0 \right\}} \qquad (23)$$

Pour l'implantation, il suffira donc pour le calcul du pas PARTAN maximal de mémoriser les deux vecteurs w_ℓ et $w_{\ell-1}$ (appelés w1 et w2 dans les algorithmes). Le calcul de $w_{\ell+1}$ se fait à l'aide des formules (19)-(21).

Algorithme "PARTAN" pour le problème d'affectation du trafic d'équilibre

0. (*initialisation*)

   $v^1 := \text{EXFLOW}\ (s_a(0))$; $\ell := 1$.

1. (*direction de descente F & W*)

   $y^\ell := \text{EXFLOW}\ (s_a(v^\ell))$

   $d^\ell := y^\ell - v^\ell$

   Si $\left| \dfrac{\sum_a s_a(v^\ell_a) d^\ell_a}{\sum_a \varepsilon_a(v^\ell_a)} \right| < \varepsilon$ alors STOP.

2. (*recherche unidimensionnelle*)

   λ_ℓ solution optimale de $\underset{0\leq\lambda\leq 1}{\text{Min}} \sum_{a\in A} S_a(v^\ell_a + d^\ell_a)$

   $z^\ell := v^\ell + \lambda_\ell d^\ell$.

3. (*recherche dans la direction PARTAN*)

Si $\ell = 1$ alors initialiser w1 et w2 (à l'aide de (19)-(20))

$$v^{\ell+1} := z^{\ell} \quad \text{(c'est-à-dire } v^2 := z^1\text{)}$$

mémoriser v^2

Sinon $d_p^{\ell} := z^{\ell} - v^{\ell-1}$

calcul du pas maximal p_{ℓ}^{max} (à l'aide de (23))

p_{ℓ} solution optimale de $\underset{0\le p\le p_{\ell}^{max}}{\text{Min}} \sum_{a\in A} S_a(v_a^{\ell-1}+pd_{pa}^{\ell})$

$$v^{\ell+1} := v^{\ell-1} + p_{\ell} d_p^{\ell}$$

mémoriser $v^{\ell+1}$ et éliminer $v^{\ell-1}$

ajuster w1 et w2 (à l'aide de (21))

$\ell := \ell + 1$; retour à 1.

On n'est plus assuré, comme dans le cas F & W, que la direction de recherche PARTAN soit de descente en z^{ℓ}. En fait, la pratique a prouvé qu'on pouvait obtenir des pas PARTAN $p_{\ell} < 1$, c'est-à-dire

$$\begin{aligned} x^{\ell+1} &= p_{\ell} z^{\ell} + (1-p_{\ell}) x^{\ell-1} \\ &= z^{\ell} + (p_{\ell}-1)(z^{\ell}-x^{\ell-1}) \\ &= z^{\ell} + \underbrace{(p_{\ell}-1)}_{<0} d_p^{\ell} \end{aligned}$$

Heureusement, la pratique a aussi montré que dans la majeure partie des cas, d_p^{ℓ} était de descente en z^{ℓ}. On peut modifier l'algorithme en forçant le pas PARTAN à être ≥ 1, c'est-à-dire en rajoutant l'instruction

$$p_{\ell} := \max(1, p_{\ell}) \quad .$$

L'avantage de cette modification est qu'alors la formule du pas maximal (23) se simplifie considérablement, comme nous l'a fait remarquer H. Spiess :

Proposition : Si $p_\ell \geq 1$, $\forall\ \ell \geq 2$ (c'est-à-dire, la direction PARTAN est de descente en z^ℓ), alors

$$p_\ell^{max} = \frac{1}{1 - \bar{\lambda}_\ell \bar{\lambda}_{\ell-1}\left(1 - \frac{\bar{p}_{\ell-1}}{p_{\ell-1}^{max}}\right)} \qquad (24)$$

(en posant : $\bar{p}_1 = 0$ et $\overline{p_1^{max}} = 1$).

Lemme : De façon générale, l'expression (24) est le premier des termes de (23) :

$$\frac{1}{1 - \bar{\lambda}_\ell \bar{\lambda}_{\ell-1}\left(1 - \frac{\bar{p}_{\ell-1}}{p_{\ell-1}^{max}}\right)} = \frac{w_{\ell-1}^0}{w_{\ell-1}^0 - \bar{\lambda}_\ell w_\ell^0} \qquad \forall\ \ell \geq 2 \qquad (25)$$

Démonstration du lemme :

par induction sur ℓ

$\ell = 2$:

$$\text{m.g. (25)} = \frac{1}{1 - \bar{\lambda}_2 \bar{\lambda}_1}$$

$$\text{m.d. (25)} = \frac{w_1^0}{w_1^0 - \bar{\lambda}_2 w_2^0} = \frac{1}{1 - \bar{\lambda}_2 \bar{\lambda}_1} \quad \text{en utilisant (19) et (20)}$$

$$\frac{1}{1-\bar{\lambda}_\ell\bar{\lambda}_{\ell-1}\left(1-\frac{\bar{p}_{\ell-1}}{1-p_{\ell-1}^{max}}\right)} = \frac{1}{1-\bar{\lambda}_\ell\bar{\lambda}_{\ell-1}\left(1-\frac{\bar{p}_{\ell-1}}{1-\frac{w^0_{\ell-2}}{w^0_{\ell-2}-\bar{\lambda}_{\ell-1}w^0_{\ell-1}}}\right)}$$

$$= \frac{1}{1-\bar{\lambda}_\ell\bar{\lambda}_{\ell-1}\left(\frac{\bar{\lambda}_{\ell-1}w^0_{\ell-1}p_{\ell-1}+\bar{p}_{\ell-1}w^0_{\ell-2}}{\bar{\lambda}_{\ell-1}w^0_{\ell-1}}\right)}$$

$$= \frac{w^0_{\ell-1}}{w^0_{\ell-1}-\bar{\lambda}_\ell\underbrace{(\bar{\lambda}_{\ell-1}w^0_{\ell-1}p_{\ell-1}+\bar{p}_{\ell-1}w^0_{\ell-2})}_{w^0_\ell \text{ par (21)}}}$$

$(\ell \geq 3)$ □

Le lemme ci-dessus indique que, de façon générale, le pas maximal trouvé par (23) est inférieur ou égal à celui trouvé par (24), l'inégalité pouvant devenir stricte comme l'ont montré les cas pratiques. Donc, dans le cas général, le pas fourni par (24) peut devenir non-réalisable. La proposition énoncée nous assure toutefois que dans le cas où $p_\ell \geq 1$, $\forall\ \ell \geq 2$, alors les pas max. fournis par (23) et (24) sont égaux.

Démonstration de la proposition :

Par le lemme, il suffit de montrer que si $p_\ell \geq 1$, $\forall\ \ell \geq 2$, alors le premier des termes de (23) est le plus petit. On va voir qu'ils sont croissants (avec i), ce qui est un peu plus.

à voir : $$\frac{w_{\ell-1}^{i}}{w_{\ell-1}^{i}-(1-\lambda_\ell)w_\ell^{i}} \le \frac{w_{\ell-1}^{i+t}}{w_{\ell-1}^{i+t}+(1-\lambda_\ell)w_\ell^{i+t}} \qquad \begin{matrix}\ell \ge 3 \\ 0\le i\le \ell-3 \\ 1\le t\le \ell-2-i\end{matrix} \qquad (*)$$

(on peut prendre $\ell \ge 3$ car pour $\ell = 2$, il n'y a qu'un seul terme dans (23), donc rien à voir!)

On peut supposer $w_{\ell-1}^{i}$ et $w_{\ell-1}^{i+t}$ strictement positifs. En effet, si $w_{\ell-1}^{j}$, $0\le j\le \ell-2$, est nul, alors w_ℓ^{j} l'est aussi.

[[Supposons $w_{\ell-1}^{j} = 0$.

Si $0\le j\le \ell-3$: $w_\ell^{j} = p_{\ell-1}(1-\lambda_{\ell-1})\underbrace{w_{\ell-1}^{j}}_{0} + (1-p_{\ell-1})w_{\ell-2}^{j}$

$$= (1-p_{\ell-1})w_{\ell-2}^{j} \quad .$$

Si $w_{\ell-2}^{j} = 0$, alors $w_\ell^{j} = 0$.

Sinon $w_{\ell-2}^{j} > 0$ et comme on veut $w_\ell^{j} \ge 0$, alors $(1-p_{\ell-1}) \ge 0$

mais $p_{\ell-1} \ge 1$ par hypothèse, donc $p_{\ell-1} = 1$, donc $w_\ell^{j} = 0$
($\ell-1 \ge 2$)

si $j = \ell-2$: $w_\ell^{j} = p_{\ell-1}(1-\lambda_{\ell-1})w_{\ell-1}^{j} = 0$]]

Donc, si $w_{\ell-1}^{i} = 0$, alors $w_{\ell-1}^{i}-(1-\lambda_\ell)w_\ell^{i} = 0$ et le terme

$\dfrac{w_{\ell-1}^{i}}{w_{\ell-1}^{i}-(1+\lambda_\ell)w_\ell^{i}}$ ne fait pas partie de (23).

Donc on peut supposer $w^i_{\ell-1} > 0$. On montre de même manière qu'on peut supposer $w^{i+t}_{\ell-1} > 0$.

L'inégalité (*) s'écrit alors :

$$\frac{1}{1 - \bar{\lambda}_\ell \dfrac{w^i_\ell}{w^i_{\ell-1}}} \leq \frac{1}{1 - \bar{\lambda}_\ell \dfrac{w^{i+t}_\ell}{w^{i+t}_{\ell-1}}}$$

ou encore : $$\bar{\lambda}_\ell \frac{w^i_\ell}{w^i_{\ell-1}} \leq \bar{\lambda}_\ell \frac{w^{i+t}_\ell}{w^{i+t}_{\ell-1}} \quad .$$

Or comme $\bar{\lambda}_\ell \geq 0$, une condition suffisante pour avoir cette inégalité est :

$$\frac{w^i_\ell}{w^i_{\ell-1}} \leq \frac{w^{i+t}_\ell}{w^{i+t}_{\ell-1}} \qquad \begin{matrix} \ell \geq 3 \\ 0 \leq i \leq \ell-3 \\ 1 \leq t \leq \ell-2-i \end{matrix} \qquad (**)$$

On va démontrer (**) par induction sur ℓ :

$\underline{\ell = 3}$: alors $i = 0$ et $t = 1$ sont les seules valeurs possibles

$$\frac{w^0_3}{w^0_2} = \frac{p_2\bar{\lambda}_2 w^0_2 + \bar{p}_2 w^0_1}{w^0_2} = p_2\bar{\lambda}_2 + \frac{\bar{p}_2}{\bar{\lambda}_1} \leq p_2\bar{\lambda}_2$$

car $p_2 \geq 1$ par hyp.

$$\frac{w^1_3}{w^1_2} = \frac{p_2\bar{\lambda}_2 w^1_2}{w^1_2} = p_2\bar{\lambda}_2$$

donc (**) est vérifiée pour $\ell = 3$.

$1 \le t < \ell-2-i$: alors $0 \le i < \ell-3$

donc, par hypothèse d'induction, on a

$$\frac{w_{\ell-1}^{i}}{w_{\ell-2}^{i}} \le \frac{w_{\ell-1}^{i+t}}{w_{\ell-2}^{i+t}}$$

ou : $$\frac{w_{\ell-2}^{i}}{w_{\ell-1}^{i}} \ge \frac{w_{\ell-2}^{i+t}}{w_{\ell-1}^{i}}$$

d'où, comme $\bar{p}_{\ell-1} \le 0$ par hypothèse,

$$\bar{p}_{\ell-1} \frac{w_{\ell-2}^{i}}{w_{\ell-1}^{i}} \le \bar{p}_{\ell-1} \frac{w_{\ell-2}^{i+t}}{w_{\ell-1}^{i+t}}$$

d'où : $$p_{\ell-1}\bar{\lambda}_{\ell-1} + \bar{p}_{\ell-1} \frac{w_{\ell-2}^{i}}{w_{\ell-1}^{i}} \le p_{\ell-1}\bar{\lambda}_{\ell-1} + \bar{p}_{\ell-1} \frac{w_{\ell-2}^{i+t}}{w_{\ell-1}^{i+t}}$$

ou : $$\frac{p_{\ell-1}\bar{\lambda}_{\ell-1}w_{\ell-1}^{i} + \bar{p}_{\ell-1}w_{\ell-2}^{i}}{w_{\ell-1}^{i}} \le \frac{p_{\ell-1}\bar{\lambda}_{\ell-1}w_{\ell-1}^{i+t} + \bar{p}_{\ell-1}w_{\ell-2}^{i+t}}{w_{\ell-1}^{i+t}}$$

ou encore, par (21) : $$\frac{w_{\ell}^{i}}{w_{\ell-1}^{i}} \le \frac{w_{\ell}^{i+t}}{w_{\ell-1}^{i+t}}$$

$t = \ell-2-i$: alors $0 \le i \le \ell-3$

on sait que $$\frac{w_{\ell-2}^{i}}{w_{\ell-1}^{i}} \ge 0$$ et comme $p_{\ell-1} \ge 1$

alors $$\bar{p}_{\ell-1} \frac{w_{\ell-2}^{i}}{w_{\ell-1}^{i}} \le 0$$

$$\text{d'où :} \quad p_{\ell-1}\bar{\lambda}_{\ell-1} + \bar{p}_{\ell-1}\frac{w^i_{\ell-2}}{w^i_{\ell-1}} \leq p_{\ell-1}\bar{\lambda}_{\ell-1}$$

$$\text{ou :} \quad \frac{p_{\ell-1}\bar{\lambda}_{\ell-1}w^i_{\ell-1} + \bar{p}_{\ell-1}w^i_{\ell-2}}{w^i_{\ell-1}} \leq \frac{p_{\ell-1}\bar{\lambda}_{\ell-1}w^{\ell-2}_{\ell-1}}{w^{\ell-2}_{\ell-1}}$$

$$\text{ou par (21) :} \quad \frac{w^i_\ell}{w^i_{\ell-1}} \leq \frac{w^{\ell-2}_\ell}{w^{\ell-2}_{\ell-1}} \qquad \square$$

On remarque que l'utilisation de la formule (24) ne nécessite plus la mémorisation des poids w_ℓ et $w_{\ell-1}$. Toutefois, le gain en espace mémoire n'est pas considérable car les vecteurs contenant ces poids ont dimension ℓ et $\ell-1$ respectivement et le nombre d'itérations de l'algorithme qu'on fait en pratique est "raisonnable". Pour utiliser la formule (24), il faut modifier l'étape 3 de l'algorithme en forçant le pas PARTAN p_ℓ à être ≥ 1, de la manière indiquée plus haut.

d) Stratégie de restriction

Pour les problèmes de grande taille, la majeure partie du temps de calcul des algorithmes précédents est consacrée à la recherche des plus courts chemins. C'est pour améliorer ce point qu'on introduit la stratégie de restriction.

Cette stratégie, décrite dans Geoffrion [9], consiste en

i) une linéarisation interne du domaine réalisable Ω de (8) (on le suppose borné) :

$$\left.\begin{array}{l} \text{Min } F\left(\sum\limits_{i\in I}\alpha_i y^i\right) \\ \text{s.à. } \sum\limits_{i\in I}\alpha_i = 1 \\ \qquad \alpha_i \geq 0, \quad i\in I \end{array}\right\} \tag{26}$$

ii) une restriction : $\alpha_i = 0$ si $i\in I-R$

Supposons $R = \{0,1,\ldots,L\}$; le problème restreint s'écrit alors :

$$\left.\begin{array}{l} \text{Min } F\left(\sum\limits_{i=0}^{L}\alpha_i y^i\right) \\ \text{s.à.: } \sum\limits_{i=0}^{L}\alpha_i = 1 \\ \qquad \alpha_i \geq 0, \quad i\in R \end{array}\right\} \tag{27}$$

En choisissant les points de base y^i, $i\in I$ d'une façon arbitrairement dense, (26) est équivalent à (8).

La stratégie de restriction consiste à résoudre (27) pour un certain ensemble R donné. Si la solution optimale ainsi trouvée satisfait à un critère d'optimalité pour (8), alors la procédure est arrêtée. Sinon on modifie l'ensemble R et on recommence. (Pour plus de détails, voir par exemple Holloway [11]).

Dans le cadre de l'algorithme de Frank & Wolfe, cette procédure peut s'exprimer ainsi :

Soit $B = \{y^i \mid i\in R\}$. $\quad |R| = L+1$

0. Trouver une solution réalisable x^1; $(x^1 \overset{\Delta}{=} y^0)$

$B := \{x^1 = y^0\}$; $L := 0$; $\ell := 1$.

1. Résoudre le sous-problème linéarisé, ce qui fournit le point extrême y^ℓ;
critère d'arrêt
ajouter y^ℓ à B ; $L := L+1$.

2. Résoudre le problème restreint (27)
soit (α_i^ℓ) la solution trouvée

$$x^{\ell+1} := \sum_{i=0}^{L} \alpha_i^\ell y^\ell \quad ; \quad \ell := \ell+1$$

retour à 1.

Il y a plusieurs façons de gérer l'évolution de l'ensemble B des points de base définissant la restriction. Dans l'approche théorique, après avoir déterminé la solution optimale de (27) à l'étape 2, on peut éliminer de B les points y^i dont le poids α_i^ℓ est devenu nul, ceci uniquement si la valeur de l'objectif a décrû strictement. Si, à l'étape 2, on élimine tous les points de B pour les remplacer par le nouveau point courant $x^{\ell+1}$, alors l'algorithme ci-dessus n'est autre que celui de Frank & Wolfe.

Pratiquement, on ne résoudra pas (27) exactement à chaque itération, mais on obtiendra une solution (α_i^ℓ) approximative. On pourrait alors éliminer de B les points y^i dont le poids α_i^ℓ est relativement très petit, imitant ainsi l'approche théorique. Nous avons préféré une méthode plus drastique qui consiste à conserver tous les nouveaux points extrémaux y^ℓ dans B jusqu'à concurrence d'un certain nombre fixe (appelé MAXPTS) et d'alors laisser tomber tous les points de B pour les remplacer par le

nouveau point courant $x^{\ell+1}$. Dans ce sens, l'algorithme adopté est une généralisation de l'algorithme de Frank & Wolfe.

Pour résoudre le problème restreint (27), on utilisera la méthode du gradient réduit (cf. Florian [5]). Dans l'implantation de l'algorithme, on pourra se borner à ne faire qu'un certain nombre fixe d'itérations restreintes (itérations du gradient réduit); cinq itérations seront effectuées dans les applications. On peut aussi terminer la recherche restreinte si le critère d'arrêt suivant est rempli :

$$\left| \frac{\nabla F(x^{\ell}) d^{\ell}}{F(x^{\ell})} \right| < \varepsilon'$$

où d^{ℓ} est la direction de descente trouvée par la méthode du gradient réduit.

L'algorithme de F & W ayant un bon comportement dans les premières itérations, on ne commencera la restriction qu'après un certain nombre (NBITER) d'itérations régulières de F & W.

L'algorithme de résolution du problème (8) prend donc la forme :

0. Trouver une solution réalisable x^1
   B:=∅ ; L:=0 ; ℓ:=1

1. Résoudre le sous-problème linéarisé ce qui fournit le point extrémal y^{ℓ}
   direction de descente F & W : $d^{\ell}:=y^{\ell}-x^{\ell}$
   critère d'arrêt de l'algorithme
   aller en 3.

2. Direction de descente du gradient réduit : d^{ℓ}
   si (critère d'arrêt de la procédure restreinte) alors aller à 4.

3. Recherche unidimensionnelle dans la direction d^{ℓ}

   soit λ_{ℓ} le pas optimal trouvé

   $x^{\ell+1} := x^{\ell} + \lambda_{\ell} d^{\ell}$

   $\ell := \ell+1$

   Si ℓ < NBITER alors retour à 1

   Si B = $\emptyset$ alors aller à 5 (c'est le cas pour la 1ère restriction)

   retour à 2.

4. Si L < MAXPTS alors aller à 6.

5. Purge de B : $B := \{x^{\ell}\}$; $L := 0$.

6. Générer un nouveau point extrémal en résolvant le sous-problème linéarisé

   $L := L+1$

   nouveau point extrémal : y^{L}

   $B := B \cup \{y^{L}\}$

   direction de descente $d^{\ell} := y^{L} - x^{\ell}$

   critère d'arrêt de l'algorithme

   retour en 3.

Les étapes {0,1,3} forment l'algorithme classique de F & W. Les étapes {2,3} résolvent le problème restreint (27).

Traduisons maintenant cet algorithme dans le cadre du problème d'équilibre (7). En référence avec le programme utilisé, appelons NEWFLO la routine qui fournit la direction de descente de la procédure du gradient réduit. La variable U représente un mode d'ajustement des α_i (cf. Florian [5] pour plus de précisions) et IT est un compteur d'itérations du gradient réduit.

Algorithme avec stratégie de restriction pour le problème d'affectation du trafic d'équilibre

0. (*initialisation*)

v^1:=EXFLOW ($s_a(0)$)

ℓ:=1; L:=0; U:=1; IT:=0;

1. (*direction de descente F & W*)

y^ℓ:=EXFLOW ($s_a(v^\ell)$)

d^ℓ:=y^ℓ-v^ℓ

si $\left| \dfrac{\sum_a s_a(v_a^\ell) d_a^\ell}{\sum_{a\in A} S_a(v_a^\ell)} \right| < \varepsilon$ alors STOP

aller à 3.

2. (*restriction*)

IT:=IT+1

NEWFLO → d^ℓ

si $\left| \dfrac{\sum_a s_a(v_a^\ell) d_a^\ell}{\sum_a S_a(v_a^\ell)} \right| < \varepsilon'$ ou si IT ≥ 5, aller à 4

3. (*recherche unidimensionnelle*)

λ_ℓ solution optimale de $\underset{0\le\lambda\le 1}{\text{Min}} \sum_{a\in A} S_a(v_a^\ell+\lambda d_a^\ell)$

$v^{\ell+1}$:=v^ℓ+$\lambda_\ell d^\ell$

ℓ:=ℓ+1

Si ℓ < NBITER alors retour à 1

Si L=0 (*première restriction*) alors aller à 5

ajuster les α_i (selon U)

U:=2

aller à 2.

4. (*réinitialisation*)

U:=1; IT:=0

Si L < MAXPTS alors aller à 6

5. (*élimination des points extrémaux générés jusqu'ici*)

éliminer les points extrémaux (s'il y en a)

mémoriser v^ℓ (la solution courante)

initialiser α_i

L:=0

6. (*génération d'un nouveau point extrémal*)

L:=L+1

y^L:=EXFLOW $(s_a(v^\ell))$

mémoriser y^L

$d^\ell := y^L - v^\ell$

si $\left| \dfrac{\sum_{a\in A} s_a(v_a^\ell) d_a^\ell}{\sum_{a\in A} s_a(v_a^\ell)} \right| < \varepsilon$ alors STOP

retour à 3.

Remarque : Pour les NBITER premières itérations, on peut penser à implanter PARTAN. C'est ce qui a été fait dans le programme donné en annexe B. Le choix de la technique F & W ou PARTAN pour ces itérations initiales est laissé à l'utilisateur (le paramètre IPARTA vaudra 0 si F & W est utilisé, 1 si c'est PARTAN).

e) Approximation quadratique

Cette méthode consiste à générer une nouvelle direction de recherche au point courant x^ℓ à l'aide de l'approximation quadratique de la fonction objectif en ce point (et non de l'approximation linéaire comme dans F & W).

Plus précisément, soit Q_ℓ l'approximation quadratique de F au point x^ℓ. L'algorithme prend alors la forme :

0. Trouver une solution réalisable x^1; $\ell := 1$

1. Résoudre le problème

$$\left.\begin{array}{l} \text{Min } Q_\ell(x) \\ \text{s.à. } Ax = b \\ \qquad x \geq 0 \end{array}\right\} \qquad (28)$$

soit z^ℓ la (une) solution optimale

$d^\ell := z^\ell - x^\ell$

2. Résoudre $\quad \underset{0 \leq \lambda \leq 1}{\text{Min}} \; F(x^\ell + \lambda d^\ell)$

soit λ_ℓ la solution optimale

$x^{\ell+1} := x^\ell + \lambda_\ell d^\ell$

$\ell := \ell + 1$

Retour à 1

(28) est un problème de programmation quadratique. L'idée de la méthode est qu'il est plus facile de résoudre (28) que le problème original. Toutes les techniques vues précédemment s'appliquent à la résolution de (28). A l'étape 1, il ne s'agit évidemment pas de résoudre exactement (28) à chaque itération. On effectuera plutôt un nombre déterminé (MAXIT) d'itérations

d'un des algorithmes F & W ou PARTAN à chaque passage dans l'étape, ce qui fournira une solution approchée de (28).

L'approximation quadratique ne devenant fidèle que près de l'optimum, et l'idée de toutes les variantes étudiées étant d'améliorer le comportement de l'algorithme de F & W près de la solution optimale, on ne commencera d'utiliser la direction fournie par l'approximation quadratique qu'après avoir effectué un certain nombre (NBITER) d'itérations régulières de F & W.

L'algorithme de résolution du problème (8) prend alors la forme (avec F & W comme sous-algorithme) :

0. (*initialisation*)
   trouver une solution réalisable x^1; $\ell := 1$

1. (*direction de descente*)
   Si $\ell \geq$ NBITER alors
      1.0 (*initialisation*)
         $z^1 := x^\ell$; $k := 1$
      1.1 (*direction de descente*)
         Résoudre le sous-problème linéaire

$$\left.\begin{array}{l} \text{Min } \nabla Q_\ell(z^k)y \\ \text{s.à.} \quad Ay = b \\ \qquad\quad y \geq 0 \end{array}\right\} \qquad (29)$$

         Soit y^k la solution optimale
         $d^k := y^k - z^k$
         Si $\left| \dfrac{\nabla Q_\ell(z^k)d^k}{Q_\ell(z^k)} \right| < \varepsilon$ alors aller à 1.3.

1.2 (*recherche unidimensionnelle*)

μ_k solution optimale de $\underset{0\le\mu\le 1}{\text{Min}}\ Q_\ell(z^k+\mu d^k)$ (30)

$z^{k+1} := z^k + \mu_k d^k$

$k := k+1$

Si $k \le$ MAXIT alors retour à 1.1

1.3 (*fin du problème quadratique*)

$y^\ell := z^k$

Sinon, résoudre le sous problème linéaire

$$\text{Min } \nabla F(x^\ell) y$$
$$\text{s.à.: } Ay = b$$
$$y \ge 0$$

Soit y^ℓ une solution optimale

$d^\ell := y^\ell - x^\ell$

Si $\left| \dfrac{\nabla F(x^\ell) d}{F(x^\ell)} \right| < \varepsilon$ alors STOP

2. (*recherche unidimensionnelle*)

λ_ℓ solution optimale de $\underset{0\le\lambda\le 1}{\text{Min}}\ F(x^\ell+\lambda d^\ell)$

$x^{\ell+1} := x^\ell + \lambda_\ell d^\ell$

$\ell := \ell+1$

retour à 1.

Traduisons maintenant cet algorithme dans le contexte du problème d'équilibre du trafic.

$$F(v^\ell) = \sum_{a\in A} \int_0^{v_a^\ell} s_a(v)dv$$

$$\frac{\partial F}{\partial v_a}(v_a^\ell) = s_a(v_a^\ell)$$

$$\frac{\partial^2 F}{\partial v_a \partial v_{a'}}(v_a^\ell) = \delta_{aa'} s'_a(v_a^\ell) \quad (s_a(v) = s_a(v_a) \Rightarrow \text{Hessien diagonal})$$

d'où : $Q_\ell(z) = F(v^\ell) + \sum_{a\in A} s_a(v_a^\ell)(z_a - v_a^\ell) + \frac{1}{2} \sum_{a\in A} s'_a(v_a^\ell)(z_a - v_a^\ell)^2$

$$\frac{\partial Q_\ell}{\partial z_a}(z) = s_a(v_a^\ell) + s'_a(v_a^\ell)(z_a - v_a^\ell)$$

Appelons $q_a^\ell(z_a) = s_a(v_a^\ell) + s'_a(v_a^\ell)(z_a - v_a^\ell)$ le "coût quadratique" associé à l'arc $a\in A$.

Le problème (29) de l'étape 1.1 s'écrit donc :

$$\left.\begin{array}{l} \text{Min} \sum_{a\in A} q_a^\ell(z_a^k) y_a \\ \text{s.à.:} \sum_{k\in K_i} h_k = \bar{g}_i, \quad i\in I \\ \qquad h_k \geq 0, \quad k\in K_i, \ i\in I \\ y_a = \sum_{i\in I} \sum_{k\in K_i} \delta_{ak} h_k \end{array}\right\} \qquad (31)$$

Le problème (31) est exactement de la forme (17). On sait donc que la solution de ce problème est une solution "tout-ou-rien" sur les plus courts chemins au sens des coûts q_a^ℓ. On pourra donc le résoudre en utilisant la même procédure EXFLOW que précédemment en y passant les coûts appropriés comme paramètres.

Il peut toutefois arriver que les coûts q_a^ℓ deviennent négatifs, impliquant la possibilité de circuits négatifs, ce qui est plutôt gênant...! En effet, $q_a^\ell(z_a) = s_a(v_a^\ell) + s_a'(v_a^\ell)(z_a - v_a^\ell)$ est l'approximation linéaire de s_a en v_a^ℓ.

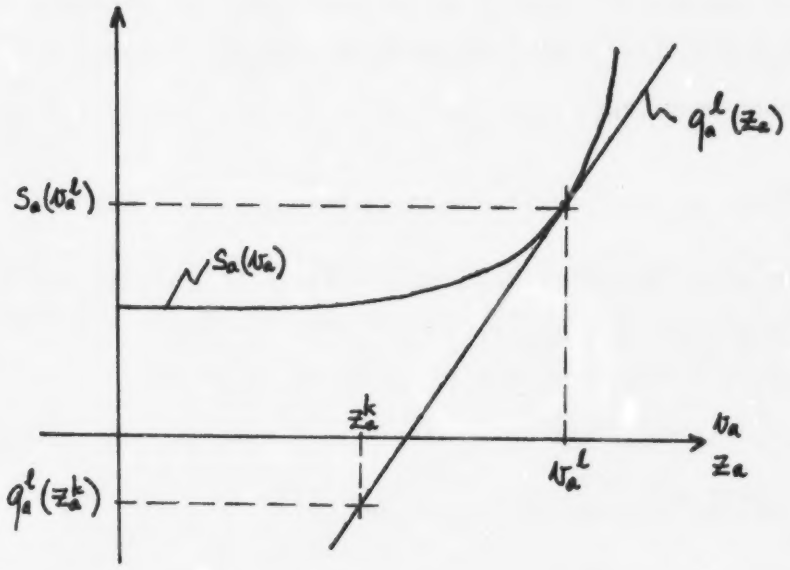

Figure 4

Si, à l'itération ℓ, l'arc a est très congestionné, la pente $s_a'(v_a^\ell)$ sera assez grande pour que la fonction de coût quadratique $q_a^\ell(z_a)$ puisse prendre des valeurs négatives pour des z_a positifs.

Il faudra donc se prévenir de ces cas dégénérés en testant le signe des coûts q_a calculés. Si un coût négatif est détecté, il faudra alors arrêter la minimisation sur l'approximation quadratique Q_ℓ et retourner sur la "vraie" fonction objectif F pour y déterminer un nouveau point $v^{\ell+1}$ ainsi qu'une nouvelle approximation quadratique $Q_{\ell+1}$.

Le problème (30) de minimisation unidimensionnelle devient ici trivial. En effet, il se résoud analytiquement :

$$0 = \frac{dQ_\ell(z^k+\mu d^k)}{d\mu} = \frac{d}{d\mu}\{Q_\ell(z^k) + \mu \nabla Q_\ell(z^k)d^k + \frac{1}{2}\mu^2 d^k HQ_\ell(z^k)d^k\}$$

$$\text{d'où} \qquad \mu_k = \frac{-\nabla Q_\ell(z^k)d^k}{d^k HQ_\ell(z^k)d^k}$$

Mais $HQ_\ell(z) = HF(v^\ell)$ définie positive, car on a supposé les coûts $s_a(v_a)$ strictement monotones.

$$\text{donc} \qquad d^k HQ_\ell(z^k)d^k > 0 \tag{32}$$

De plus, lorsqu'on entre dans l'étape 1.2, y^k est une solution optimale de (29); on a donc :

$$\nabla Q_\ell(z^k)y^k \leq \nabla Q_\ell(z^k)z^k$$

car z^k est une solution réalisable de (29)

$$\text{donc} \qquad -\nabla Q_\ell(z^k)d^k \geq 0$$

d'où, avec (32) : $\mu_k \geq 0$.

Comme $HQ_\ell(z)$ est définie positive, Q_ℓ est strictement convexe et alors la solution de (30) est fournie par

$$\mu_k = \min \left\{1, -\frac{\nabla Q_\ell(z^k)d^k}{d^k HQ_\ell(z^k)d^k}\right\}$$

ou encore :

$$\mu_k = \min \left\{1, -\frac{\sum_{a\in A} [\, s_a(v_a^\ell) + s_a'(v_a^\ell)(z_a^k - v_a^\ell)\,]\, d_a^k}{\sum_{a\in A} s_a'(v_a^\ell)(d_a^k)^2}\right\} \tag{33}$$

La résolution implicite du problème de recherche unidimensionnelle dans le cas de l'approximation quadratique rend cette technique intéressante, surtout qu'au voisinage de la solution optimale du problème convexe, cette approximation est fidèle. Elle le sera à tel point dans les applications que la recherche unidimensionnelle de l'étape 2 s'avérera inutile. On aura ainsi supprimé toute minimisation unidirectionnelle de l'algorithme (excepté dans les NBITER premières itérations régulières de F & W).

On pourra d'autre part espérer encore gagner en efficacité en implantant PARTAN à la résolution du sous-problème quadratique (28) car cette méthode est dans ce cas de type "directions conjuguées". Le pas dans la direction PARTAN se calculera avec une formule identique à (33), sauf qu'on lui permettra d'être plus grand que 1. On le forcera en fait à se trouver dans l'intervalle $[0,p_k^{max}]$, où p_k^{max} sera calculé à l'aide de (23).

Algorithme avec approximation quadratique pour le problème d'affectation du trafic d'équilibre

0. (*initialisation*)

   v^1:=EXFLOW($s_a(0)$); ℓ:=1

1. (*direction de descente*)

   Si $\ell \geq$ NBITER alors (*résolution du problème quadratique*)

   1.0 (*initialisation*)

   z^1:=v^ℓ; k:=1

   1.1 (*direction de descente*)

   Si $q_a(z^k) < 0$ pour un $a \in A$ alors aller à 1.3.

y^k:=EXFLOW $(q_a(z^k))$

d^k: $y^k - z^k$

Si $\left| \dfrac{\sum_{a\in A} q_a^\ell(z_a^k) d_a^k}{Q_\ell(z^k)} \right| < \varepsilon$ alors aller à 1.3

1.2 (*pas optimal*)

z^{k+1}:=$z^k + \mu_k d^k$ où μ_k obtenu de (33)

k:=k+1

Si k ≤ MAXIT retour à 1.1

1.3 (*fin du problème quadratique*)

y^ℓ:=z^k

Sinon y^ℓ:=EXFLOW $(s_a(v_a^\ell))$

d^ℓ:=$y^\ell - v^\ell$

si $\left| \dfrac{\sum_{a\in A} s_a(v_a^\ell) d_a^\ell}{\sum_{a\in A} S_a(v_a^\ell)} \right| < \varepsilon$, alors STOP

2. (*recherche unidimensionnelle*)

λ_ℓ solution optimale de $\underset{0\le\lambda\le 1}{\text{Min}} \sum_{a\in A} S_a(v_a^\ell + \lambda d_a^\ell)$

$v^{\ell+1}$:=$v^\ell + \lambda_\ell d^\ell$

ℓ:=ℓ+1

retour à 1.

4. RÉSEAUX TESTS

Les tests ont été effectués sur trois réseaux de tailles caractéristiques.

a) Petit réseau

Ce réseau imaginaire est tiré de [16]. On l'appellera "réseau Dupuis" dans la suite en référence à cet article.

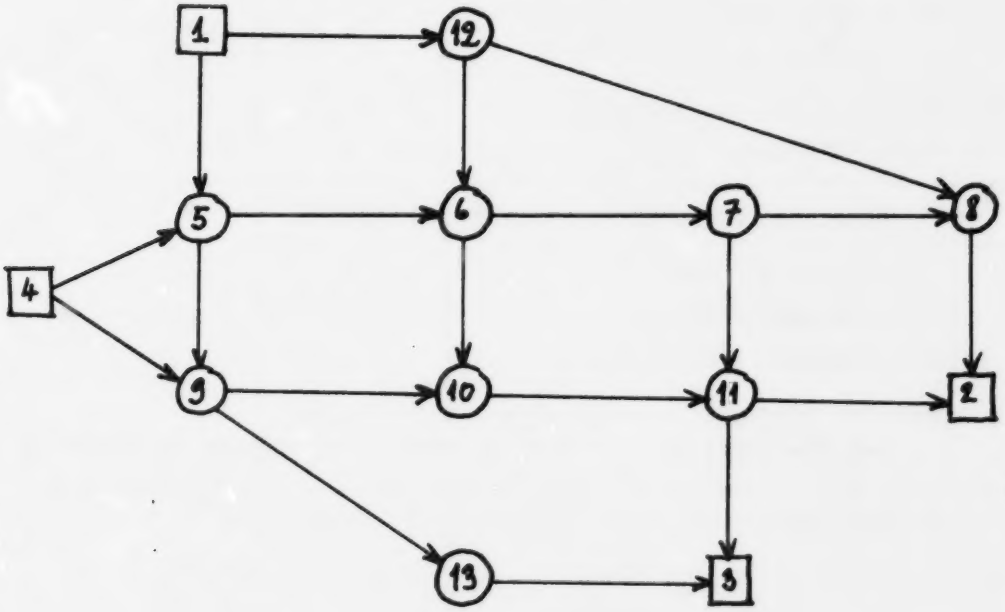

Figure 5

nombre de noeuds : NNODES = 13
nombre d'arcs : NLINK = 19
nombre de centroïdes : NCENT = 4

Les fonctions de coût sont linéaires, à savoir :

$$s_a(v_a) = \alpha_a v_a + \beta_a \ , \qquad \alpha_a, \beta_a \text{ constantes .}$$

b) Réseau moyen

C'est le réseau de la ville de Hull.

NNODES = 501
NLINK = 798
NCENT = 23

Les fonctions de coût sont plus complexes (BPR) et dépendent de plusieurs paramètres. En gros, elles sont du type :

$$s_a(v_a) = s_a(0)\left(1+\alpha_a\left(\frac{v_a}{\gamma_a}\right)^{\beta_a}\right) \qquad \alpha_a, \beta_a, \gamma_a \text{ constantes}$$

c) Grand réseau

C'est le réseau de la ville de Winnipeg

NNODES = 1052
NLINK = 2836
NCENT = 147

Les fonctions de coût sont du même type que pour le réseau de Hull.

5. RÉSULTATS ET COMMENTAIRES

Si les variantes F & W régulier et PARTAN sont bien définies, ce n'est pas le cas des autres algorithmes présentés où des paramètres doivent être choisis :

méthode	paramètre à choisir	signification du paramètre
heuristique	C	coefficient modifiant le pas F & W (on le choisira plus grand que 1)
	REG	nombre d'itérations régulières de F & W entre deux "faux" pas
restriction	NBITER	nombre d'itérations initiales sans restriction
	IPARTA	0 : itérations initiales avec F & W 1 : itérations initiales avec PARTAN
	MAXPTS	nombre maximal de points extrémaux retenus dans la restriction
approximation quadratique	NBITER	nombre d'itérations régulières de F & W
	IPARTA	0 : F & W est utilisé pour résoudre le sous-problème quadratique 1 : PARTAN est utilisé
	MAXIT	nombre d'itérations maximal sur une approximation quadratique

Tableau 1

Remarque : De façon générale, lorsqu'on parlera du nombre d'itérations d'un algorithme, on entendra le nombre de fois que la sous-routine EXFLOW de calcul des plus courts chemins est appelée.

a) Réseau Dupuis

La meilleure borne inférieure pour la fonction économique obtenue au cours des essais (grâce à l'inéquation (9)) est MBINF = 85028,072.

Cette valeur ayant été atteinte (par PARTAN et restriction), on peut la considérer comme la valeur optimale du problème :

OPT = 85028,072

1) Heuristique

La figure 6 fournit une idée du choix judicieux de la constante de perturbation C. Tous les essais dont les résultats sont fournis dans cette figure comportent cinq itérations initiales régulières de F &W, puis le pas F &W est faussé à chaque itération (REG=0).

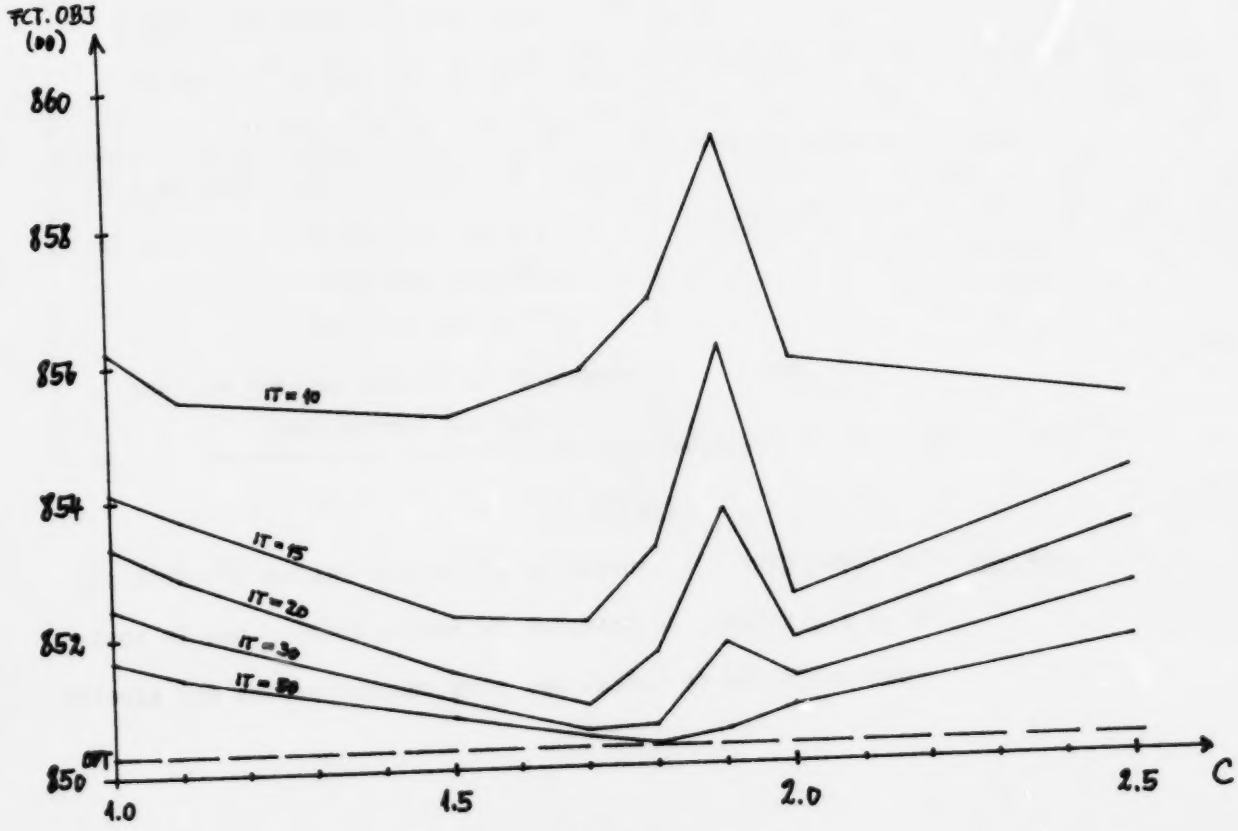

Figure 6

On conclut que C = 1,8 semble être le meilleur choix. On a alors testé différentes valeurs pour REG en conservant cette valeur optimale pour C. Le tableau 2 fournit les valeurs de la fonction objectif à différentes étapes.

# d'itérations \ REG	0	1	2	4	6
10	85680	85536	85443*	85680	85488
20	85164	85147*	85236	85294	85296
30	85059*	85112	85177	85202	85196
50	85031*	85083	85118	85134	95136
100	85030*	85059	85072	85034	85037

* meilleur résultat pour un nombre donné d'itérations

Tableau 2

Le premier choix qui consistait à effectuer un pas heuristique à chaque itération se révèle être le bon. Pour les comparaisons avec les autres méthodes, nous retiendrons donc la version REG=0, C=1,8.

Comme on peut le constater sur la figure 6, pour $1,0 < C \leq 2,2$, la vitesse de convergence de l'algorithme de F &W (C=1,0) est améliorée (le comportement étrange pour C=1,9 se normalisant lorsque le nombre d'itérations augmente). On peut immédiatement conclure de cette observation qu'il est inutile de dépenser trop d'efforts dans la précision de la recherche unidimensionnelle. (En l'occurence, pour des coûts linéaires, cela n'a pas d'importance vu que le problème de minimisation unidirectionnel se résoud analytiquement!)

2) Restriction

Des tests ont été réalisés avec les valeurs suivantes des paramètres :

NBITER = 5, 10, 15, 20
MAXPTS = 6, 11, 16

(MAXPTS = n+1 = nombre de points extrémaux retenus plus 1 pour le point courant)

Dans chaque cas on a testé les deux variantes IPARTA égale 0 ou 1 (sans ou avec PARTAN dans les NBITER premières itérations).

Les résultats trouvés démontrent que le nombre de points extrêmes retenus pour la restriction (MAXPTS) ne joue aucun rôle dans la qualité de la convergence. Ceci est dû au fait que le poids des différents points extrêmes retenus décroît très rapidement avec leur nombre. Seuls les trois ou quatre premiers points retenus possèdent des poids significatifs, d'où l'inutilité d'en conserver trop. Nous ne nous intéresserons donc ici qu'aux résultats obtenus avec MAXPTS=6.

Le tableau 3 fournit les valeurs de la fonction économique (on ne donne que les centaines, la valeur des milliers étant toujours 85) en fonction du nombre d'itérations. PARTAN étant la méthode la plus efficace au début (cf. figure 9), c'est évidemment lorsque le nombre d'itérations initiales NBITER est grand et que PARTAN est utilisé qu'on obtient les meilleurs résultats. Cependant, après un certain nombre d'itérations (ce nombre augmentant avec NBITER), ce sont les alternatives utilisant F & W dans les premières itérations qui prennent l'avantage. Le choix NBITER=5

# d'itérations \ NBITER	5		10		15		20	
10	785	380	620	120*	620	120*	620	120*
11	581	334	556	112*	556	114	556	114
13	217	277	528	109	464	105*	464	105*
15	085*	261	501	106	412	099	412	099
20	041*	136	426	094	356	097	336	080
30	038	107	089	078	033*	085	274	077
40	037	093	072	075	028*	080	060	075
50	036	081	068	073	028*	078	052	073
IPARTA	0	1	0	1	0	1	0	1

* meilleur résultat pour un nombre donné d'itérations

Tableau 3

fournit alors le plus vite un bon comportement. Toutefois, près de l'optimum, c'est le choix NBITER=15 qui s'avère être le meilleur.

Ces résultats sont confirmés par le tableau 4 qui fournit le temps CPU (en secondes) nécessaire à chaque version pour atteindre un "gap" donné. Le "gap" considéré ici est le suivant :

$$GAP = \frac{\text{(valeur de la fct. obj.) - (valeur de la borne inf. courante)}}{\text{valeur de la fonction objectif}}$$

La figure 7 donne une idée plus précise du comportement des deux meilleures variantes NBITER=5 et 15. Cette deuxième version est, comme nous l'avons remarqué, efficace près de l'optimum. Elle atteint la borne inférieure 85028,072 avec trois décimales exactes après seulement 53 itérations (1,63 CPU-secondes). Cependant, la première version fournit un "bon" comportement plus rapidement que la deuxième. Nous la retiendrons donc pour les comparaisons subséquentes.

3) Approximation quadratique

Les fonctions de coût étant linéaires, l'approximation quadratique de la fonction économique est la fonction elle-même. Cette technique n'apporte donc rien dans ce cas.

4) Comparaisons

Comparons maintenant les quatre versions retenues. Les figures 8 et 9 donnent l'évolution de la valeur de la fonction économique en fonction du nombre d'itérations et du temps de calcul respectivement. Les temps

NBITER	5		10		15		20	
GAP < 1,00%	0,80	0,72	0,59	0,44*	0,46	0,44*	0,46	0,44*
GAP < 0,10%	0,97*	2,04	1,57	1,23	1,07	1,76	1,48	1,30
GAP < 0,05%	1,54	> 6	4,25	5,78	1,31*	5,81	1,66	5,89
GAP < 0,01%	>6	> 6	> 6	> 6	1,56*	> 6	> 6	> 6
IPARTA	0	1	0	1	0	1	0	1

* meilleurs temps dans une ligne

Tableau 4

considérés étant très courts (0-1,5 secondes), ils sont relativement peu fiables : ils peuvent varier sensiblement lors de deux essais d'un même algorithme. La représentation en fonction du nombre d'itérations est donc plus appropriée dans ce cas.

La courbe obtenue avec la stratégie de restriction possède la forme rêvée pour un algorithme de résolution du problème d'équilibre. Toutefois, PARTAN est plus rapide au tout début et, aux alentours de l'itération 60, une direction de descente miracle a permis à cette méthode d'atteindre l'optimum, battant ainsi toutes les autres.

Les trois méthodes présentent toutes une nette amélioration sur l'algorithme de F & W. Après 65 itérations, on peut dire qu'elles s'équivalent.

b) Réseau de Hull

La meilleure borne inférieure pour la fonction économique obtenue au cours des essais est

MBINF = 34 807,65

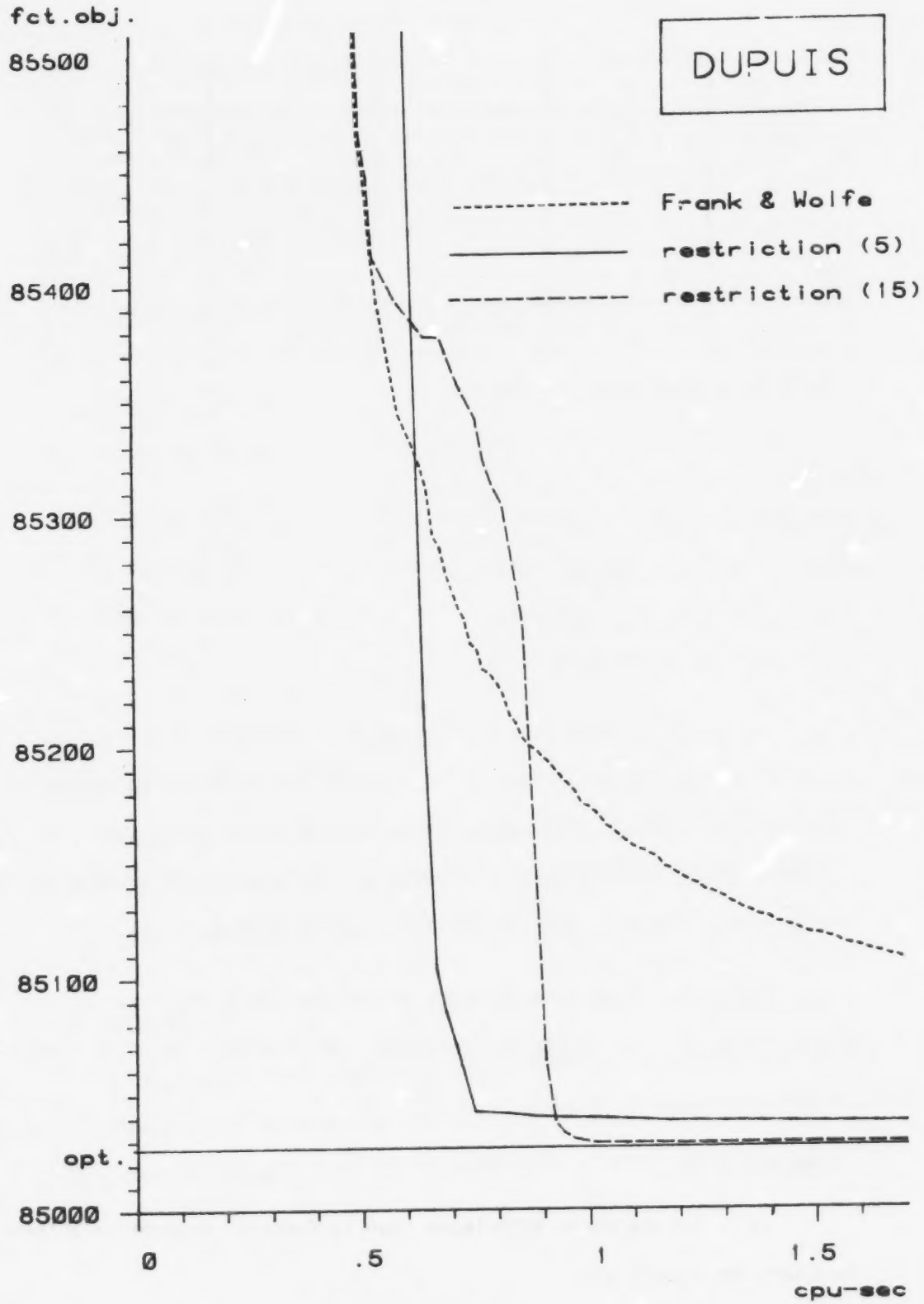
fct.obj.
85500
85400
85300
85200
85100
opt.
85000
0
.5
1
1.5
cpu-sec
DUPUIS
Frank & Wolfe
restriction (5)
restriction (15)

Figure 7

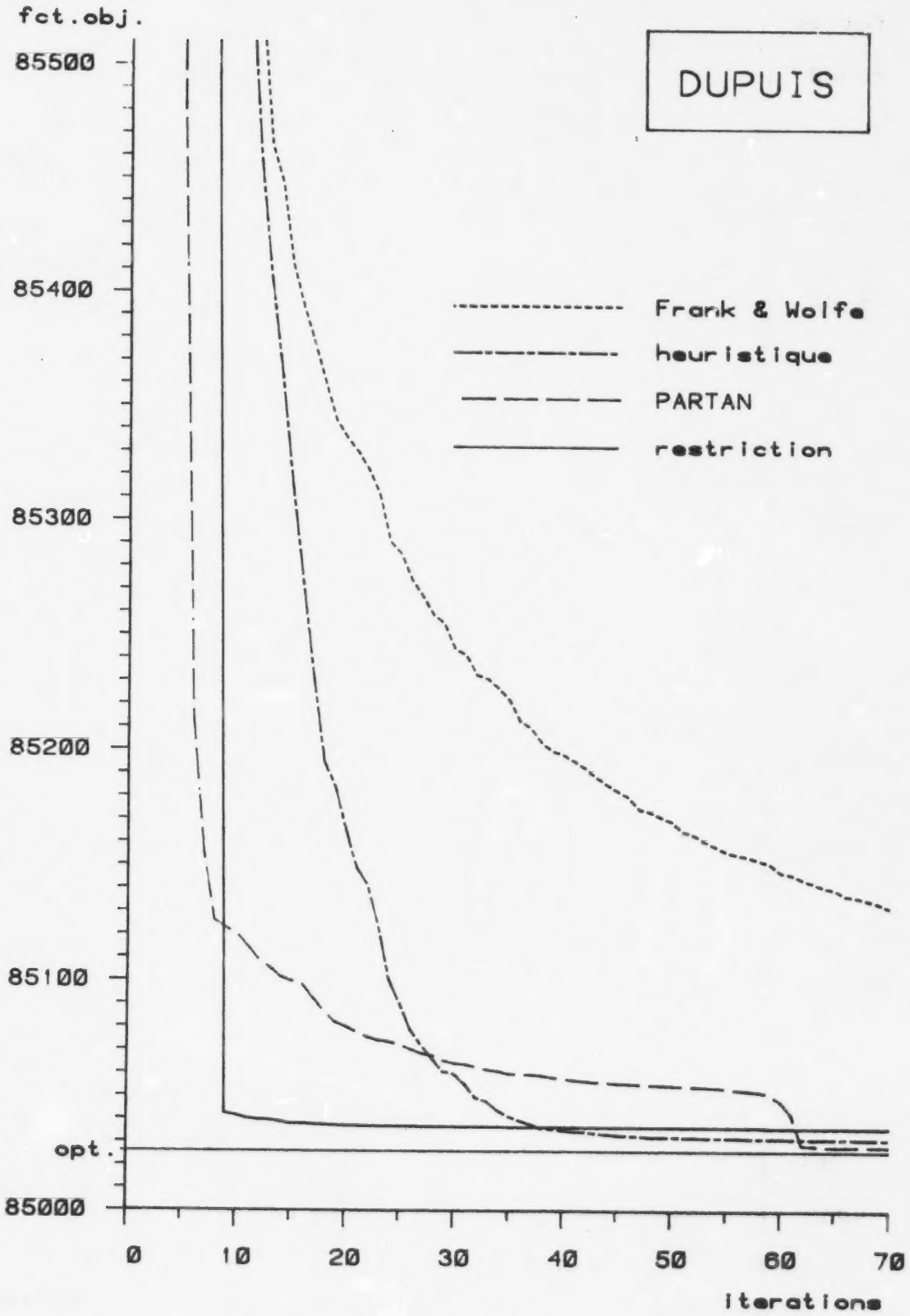
fct.obj.
DUPUIS
85500
85400
85300
85200
85100
opt.
85000
0
10
20
30
40
50
60
70
iterations
Frank & Wolfe
heuristique
PARTAN
restriction

Figure 8

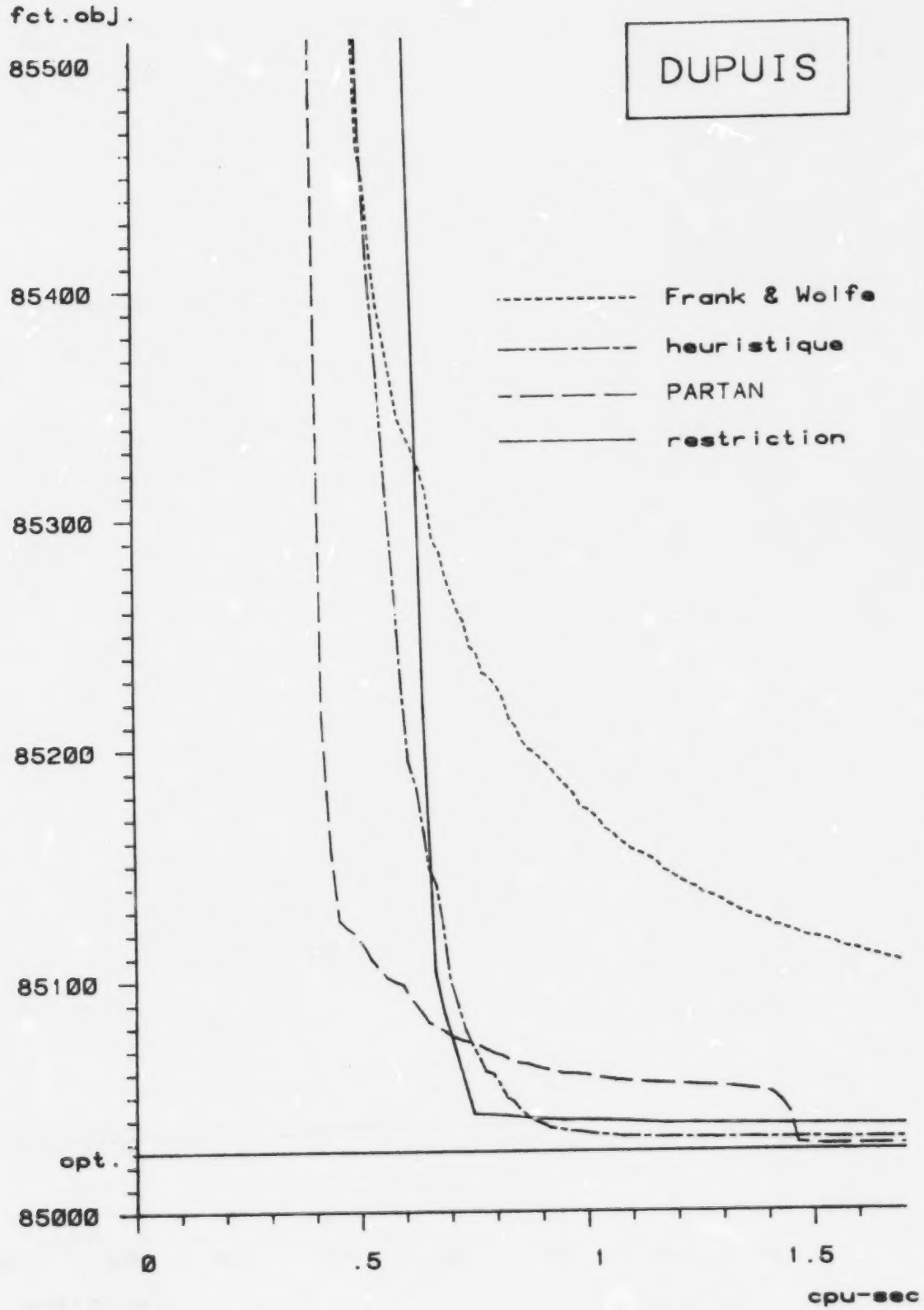
fct.obj.
85500
85400
85300
85200
85100
opt.
85000
0
.5
1
1.5
cpu-sec
DUPUIS
Frank & Wolfe
heuristique
PARTAN
restriction

Figure 9

1) Heuristique

Des tests similaires à ceux effectués pour le réseau Dupuis ont été faits pour Hull. On a commencé par tester le paramètre REG pour la valeur de C obtenue précédemment, c'est-à-dire C=1,8. Le tableau 5 fournit l'état de la fonction objectif après différents temps de calcul (CPU)

CPU-SEC \ REG	0	1	4
20	35198	35118	35097*
30	35069	34920*	34949
40	35003	34869*	34885
50	34965	34846*	34862
70	34906	34825*	34834
100	34865	34817*	34823

* meilleur résultat pour un temps donné

Tableau 5

On a donc retenu REG=1 (c'est-à-dire faire alterner les bons et les faux pas de F&W) pour la suite des analyses. On a alors retesté le paramètre C et obtenu la figure 10. Les comportements en dents de scie se sont accentués par rapport à ceux trouvés pour le réseau Dupuis. Le choix d'une meilleure valeur de C pour le début de l'algorithme est donc plutôt aléatoire. Après 100 secondes, la situation se régularise et on retrouve la valeur optimale de C=1,8. C'est donc cette version qu'on retiendra pour les comparaisons.

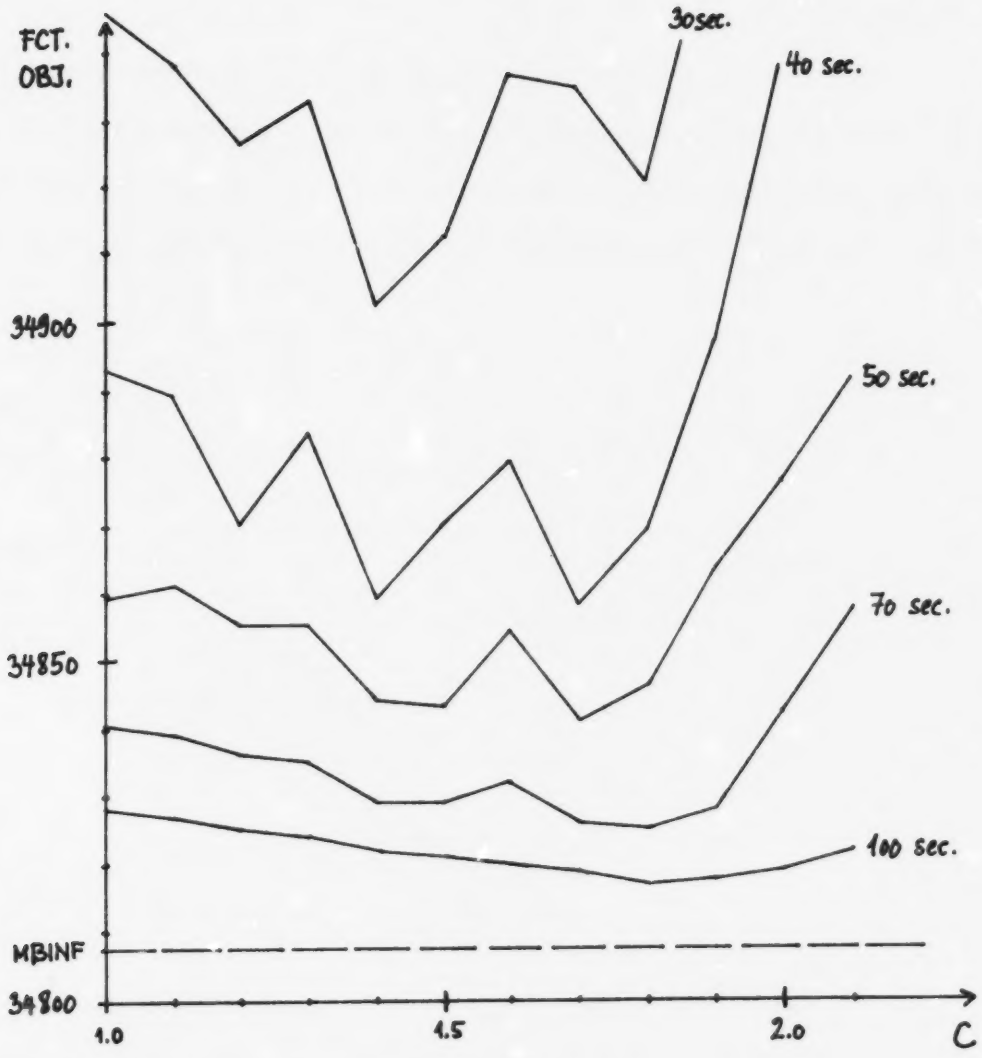

Figure 10

2. Partan

Comme on l'a vu au chapitre 3, on a deux façons de calculer le pas maximal dans la direction PARTAN (formules (23) et (24)); dans le cas de (24), on doit supposer le pas plus grand ou égal à 1, donc le forcer à l'être s'il ne l'est pas. Pour utiliser la formule (24), on peut aussi,

dès qu'on rencontre un pas PARTAN plus petit que 1, recommencer le processus (réoptimisation) à partir du point courant, c'est-à-dire refaire deux itérations de F&W. Voici une comparaison de ces différentes formules. Le tableau 6 fournit la valeur de la fonction économique en différents nombres d'itérations ainsi que le temps CPU (secondes) pour y parvenir.

On peut conclure de ce tableau que le temps gagné par l'utilisation de la formule (24) au lieu de (23) est nul. Par contre, on remarque que la fonction économique décroît plus vite lorsque le pas est forcé à être plus grand ou égal à 1, c'est-à-dire lorsqu'on ne prend pas toujours le pas exact. Ce comportement peut se rapprocher de celui de l'algorithme heuristique de Weintraub où l'on améliore la vitesse de convergence de l'algorithme de F&W en faussant le pas F&W exact.

La version de réoptimisation lorsqu'un pas plus petit que 1 était détecté n'a pas donné de meilleurs résultats. On peut penser réoptimiser à intervalles réguliers d'itérations de l'algorithme, comme le présentent Bazaraa et Shetty [2]. Nous avons alors testé différentes valeurs pour la longueur de cet intervalle. Le tableau 7 fournit les valeurs de la fonction économique à différentes étapes ainsi que le temps CPU pour y parvenir. La meilleure version est celle où l'on réoptimise toutes les quatre itérations. Nous avons ensuite comparé cette dernière version avec celles sans réoptimisation (formule (23) avec pas ≥ 0 et formule (24) avec pas plus grand ou égal à 1). La figure 11 montre que la version réoptimisée possède un meilleur comportement au début, mais qu'au voisinage de l'optimum, ce sont les versions non-réoptimisées qui l'emportent. Nous retiendrons donc la meilleure de ces dernières versions pour les comparaisons.

	pas maximal calculé avec la formule (23)		pas maximal calculé avec la formule (24)	
valeurs acceptées pour le pas PARTAN	$[0,p^{max}]$	$[1,p^{max}]$	$[1,p^{max}]$	réopt. si pas < 1
10	34954.27 26.04	34876.63 26.10	34876.63 26.04	34945.05 24.82
20	34835.38 52.24	34825.23 52.10	34825.23 52.04	34861.05 51.35
30	34816.62 77.29	34816.51 76.30	34816.51 76.25	34828.67 76.66
40	34814.25 101.88	34811.97 100.69	34811.97 100.58	34815.54 99.78
50	34812.60 124.96	34811.26 124.29	34811.26 124.18	34812.78 122.44
70	34811.04 170.97	34809.68 168.90	34809.68 168.74	34810.66 167.58
100	34809.78 237.95	34809.25 237.17	34809.25 237.03	—

Tableau 6

	nombre d'itérations entre deux réoptimisations				
nombre d'itérations	3	4	5	6	9
10	34966.44 22.34	34851.29 25.58	34848.22* 23.73	35052.37 24.62	34954.27 26.00
20	34851.27 42.36	34819.57* 45.90	34820.61 46.70	34860.54 48.64	34860.63 49.22
30	34834.46 63.20	34816.46* 66.39	34817.18 65.80	34836.83 72.15	34831.62 71.29
40	34824.57 82.59	34813.93* 86.47	34814.79 85.69	34822.17 94.87	34824.55 93.31
en 100 secondes	34820.70	34813.38*	34814.02	34820.53	34821.94

* meilleur résultat pour une ligne

Tableau 7

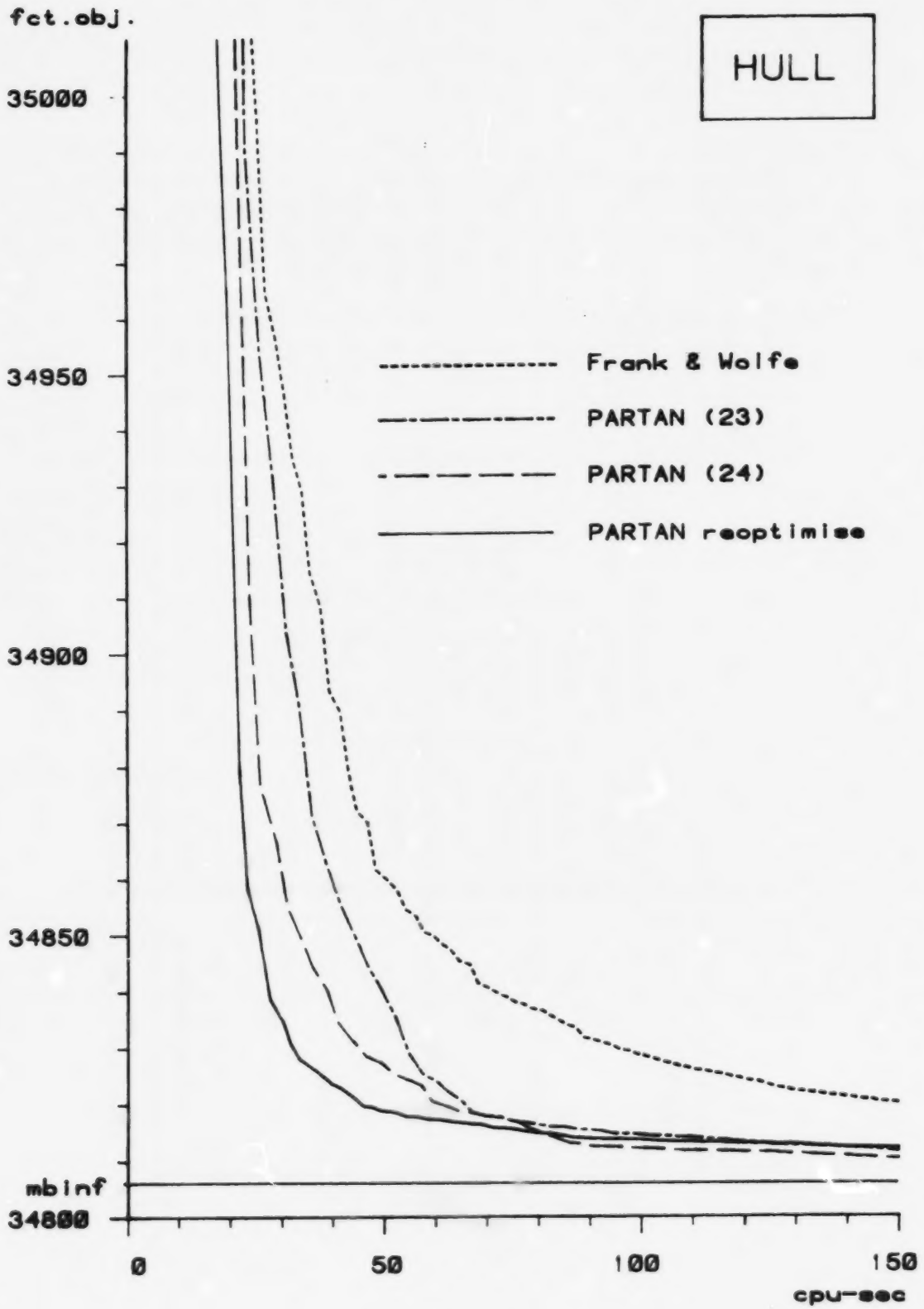
fct.obj.
HULL
35000
34950
34900
34850
mbinf
34800
0
50
100
150
cpu-sec
Frank & Wolfe
PARTAN (23)
PARTAN (24)
PARTAN reoptimise

Figure 11

3. Restriction

Des tests ont été effectués avec les valeurs suivantes des paramètres concernés dans cet algorithme :

NBITER = 5, 10, 15
MAXIT = 6, 11, 16.

Le tableau 8 fournit les temps nécessaires aux différentes variantes pour atteindre un gap < 0,1%.

MAXIT / NBITER	6		11		16	
5	75,02	46,83*	82,50	52,24	89,17	53,09
10	78,65	70,56	89,92	84,41	93,32	71,28
15	86,07	78,26	75,31	80,36	76,58	79,87
IPARTA	0	1	0	1	0	1

Tableau 8

Comme dans le cas du réseau Dupuis, c'est la version NBITER=5 et MAXIT=6 qui s'est révélée être la meilleure, à la différence près que c'est, cette fois, en utilisant PARTAN dans les premières itérations.

Nous avons voulu ici cerner un peu mieux le rôle du nombre de points extrêmes retenus lorsque ce nombre est relativement petit, c'est-à-dire lorsque tous les poids de ces points extrêmes sont significatifs. Par la même occasion, nous avons réalisé une analyse un peu plus fine du paramètre NBITER. Le tableau 9 fournit pour chaque version la valeur atteinte

NBITER \ MAXPTS	4	5	6	7
3	34825.98 >100	34823.10 85.35	34822.76 84.59	34818.96 75.31
4	34825.64 93.19	34820.73 77.38	34820.19 66.35	34822.22 79.34
5	34823.12 86.97	34812.96 54.23	34812.68 46.83	34813.44 52.08
6	34822.59 76.89	34821.41 67.92	34820.66 70.53	34820.17 63.70

Tableau 9

par la fonction objectif après 100 secondes de temps CPU ainsi que le temps nécessaire pour atteindre un gap plus petit que 0,1%. Ce tableau démontre qu'avec l'expérience on acquiert un fin nez : la même version NBITER=5, MAXPTS=6, ressort vainqueur. C'est donc celle-là que nous conserverons pour les comparaisons.

4. Approximation quadratique

Les paramètres à tester dans ce cas sont :

NBITER = nombre d'itérations initiales de F&W régulier

MAXIT = nombre maximal d'itérations sur une approximation quadratique

IPARTA = { 1 si PARTAN est utilisé pour résoudre le sous-problème quadratique
         { 0 si c'est F&W qui est utilisé.

NBITER / CPU-SEC	5		10	
20	34971	35016	35051	35050
30	887	34884	34901	34885
40	856	848	864	839
50	840	833	846	829
100	821	815	822	815
170	815	811	815	811
IPARTA	0	1	0	1

Tableau 10

On va d'abord tester ce dernier paramètre avec NBITER=5,10 et MAXIT=4. Comme on le constate dans le tableau 10, l'utilisation de PARTAN est nettement plus efficace (ce qui ne nous étonne pas...!) et nous garderons IPARTA à 1 pour les essais qui suivront.

Du tableau 11, on déduit que le nombre d'itérations sur l'approximation quadratique ne joue pas un grand rôle dans la vitesse de convergence (cf. ligne "fonction objectif après 170 secondes") alors qu'il a plutôt tendence à ralentir cette vitesse au début de l'algorithme. Ce fait est confirmé par les temps pris pour atteindre un certain gap.

NBITER	5				10			
MAXIT	4	6	8	10	4	6	8	10
fct.obj. après 30 sec.	34884	34870*	34884	34894	34885	34890	34898	35056
fct.obj. après 170 sec.	34811	34810*	34810*	34810*	34811	34810*	34810*	34810*
GAP < 1%	15.67*	18.23	23.72	36.60	24.33	26.97	29.62	32.24
GAP < .1%	36.45*	44.16	44.57	49.68	39.83	42.61	50.48	58.38
GAP < .01%	72.82	75.52	86.44	102.05	71.00*	73.91	92.32	97.60

* meilleur résultat dans une ligne donnée

Tableau 11

Ici :
$$GAP = \frac{\sum_{a \in A} s_a(v_a^\ell) d_a^\ell}{\sum_{a \in A} S_a(v_a^\ell)} \quad (13) \quad = \quad \frac{\Delta^\ell}{F(v^\ell)}$$

L'idée est alors de faire augmenter le nombre d'itérations à effectuer sur l'approximation quadratique au fur et à mesure que celle-ci devient bonne, c'est-à-dire au fur et à mesure qu'on se rapporche de l'optimum. Pour les essais suivants, nous utiliserons donc un paramètre MAXIT variable de la forme

$$MAXIT \triangleq 4 + 1(X)$$

c'est-à-dire qu'on augmentera MAXIT de 1 à toutes les X itérations (itérations sur la vraie fonction).

Du tableau 11, on peut conclure qu'il est généralement meilleur de n'effectuer que cinq itérations régulières de F & W plutôt que dix. C'est pourquoi les tests dont les résultats sont donnés dans le tableau 12 ont tous été effectués avec NBITER=5.

De ce dernier tableau, on retire que le GAP n'est pas forcément une bonne mesure de la décroissance de la fonction économique, puisque la version 4 + 1(2) fournit les meilleurs résultats en ce qui concerne la la valeur de la fonction économique, mais pas au niveau du GAP. Les valeurs fournies dans ce tableau sont calculées par interpolation linéaire des valeurs accessibles à chaque vraie itération ℓ (cf. algorithme p. 39).

Nous retiendrons donc la version NBITER=5, MAXIT=4+1(2), IPARTA=1 pour les comparaisons qui suivent.

MAXIT	4	4 + 1(1)	4 + 1(2)	4 + 1(3)	4 + 1(5)
30 sec.	34883.58	34856.18*	34860.47	34858.84	34883.62
40 sec.	847.73	828.82	822.98*	825.58	847.80
100 sec.	814.75	811.99	811.32*	811.87	813.68
170 sec.	810.66	809.80	809.63	809.50*	809.73
GAP < .1%	36.45*	48.25	42.94	41.62	36.45*
GAP < .01%	72.82*	103.33	82.11	75.57	92.55

* meilleur résultat dans une ligne donnée

Tableau 12

5. Comparaisons

Les figures 12 et 13 donnent l'évolution de la fonction économique en fonction du nombre d'itérations et du temps de calcul respectivement. La remarque faite quant à la précision relative des temps de calcul pour le réseau Dupuis ne tient plus ici.

La représentation en fonction du nombre d'itérations fournit une idée de la qualité des directions de descente générées par les différents algorithmes. C'est la technique de restriction qui donne les meilleurs résultats, suivie de près par la technique PARTAN, cette dernière s'avérant être la plus efficace proche de l'optimum.

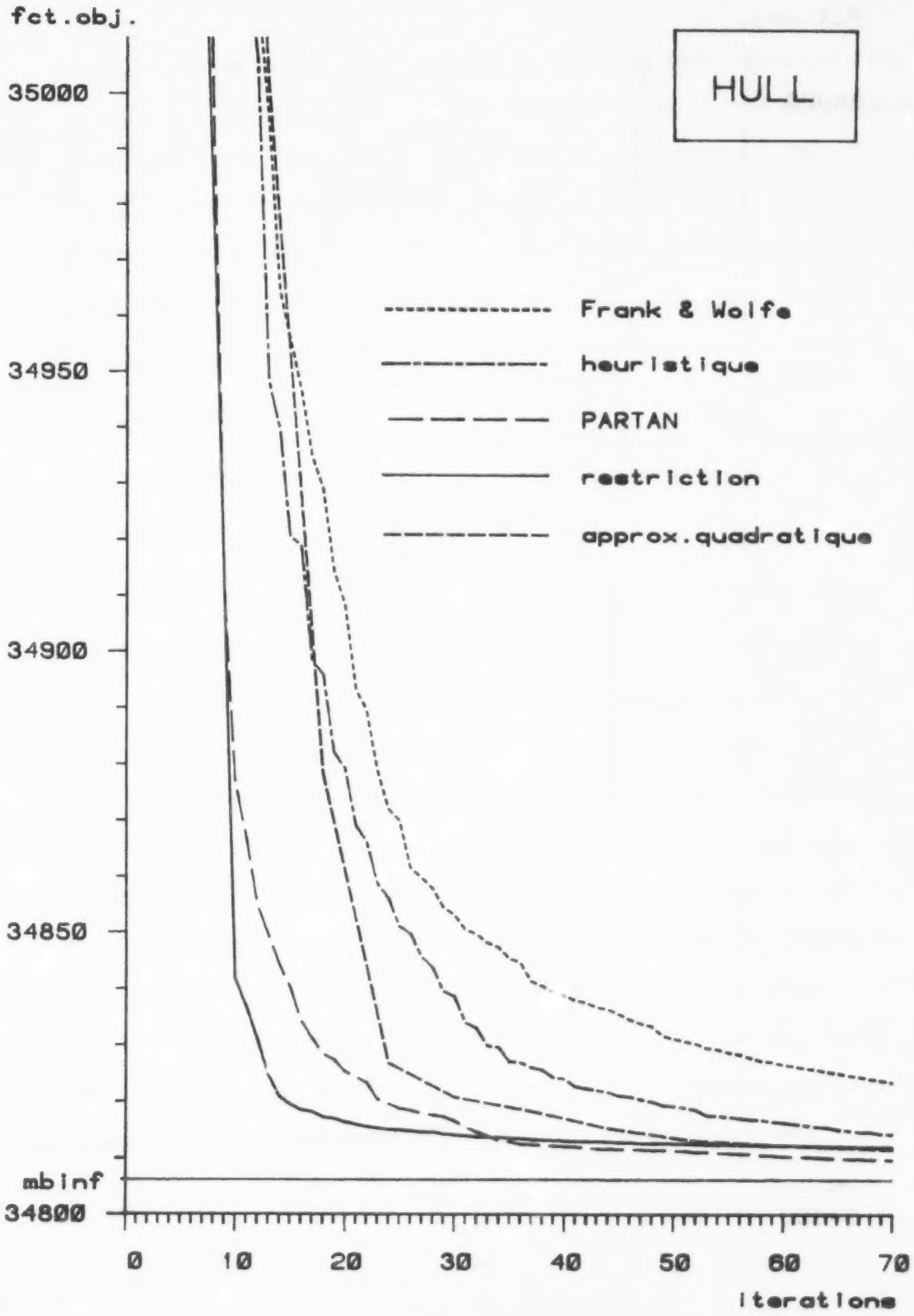
fct.obj.
HULL
35000
34950
34900
34850
mbinf
34800
Frank & Wolfe
heuristique
PARTAN
restriction
approx.quadratique
0
10
20
30
40
50
60
70
iterations

Figure 12

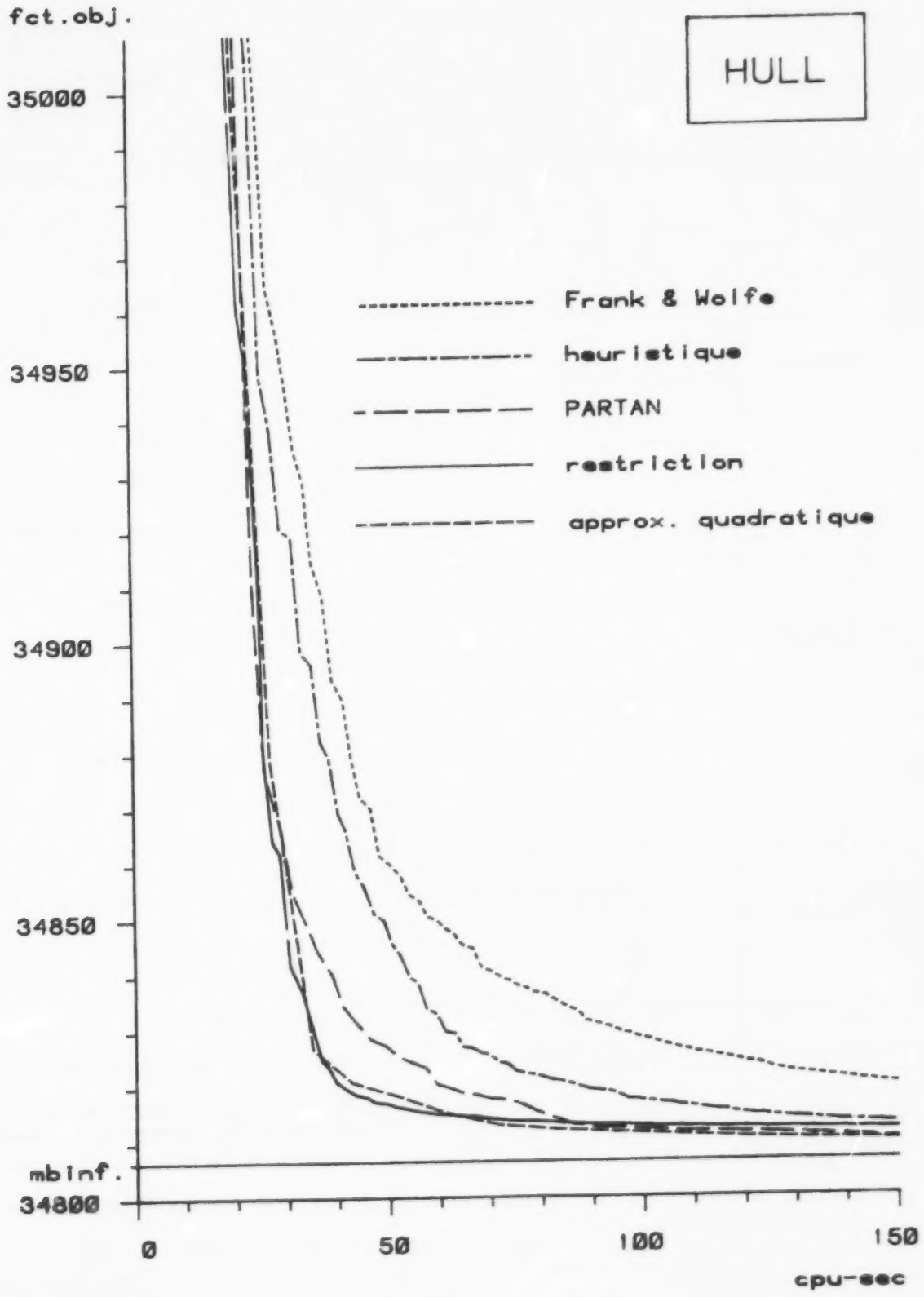
fct.obj.
HULL
35000
34950
34900
34850
mbinf.
34800
0
50
100
150
cpu-sec
Frank & Wolfe
heuristique
PARTAN
restriction
approx. quadratique

Figure 13

En ce qui concerne l'efficacité au niveau du temps de calcul (fig. 13), c'est la méthode d'approximation quadratique qui fournit généralement les meilleurs résultats, quoiqu'ils soient fort comparables à ceux obtenus avec la restriction ou avec PARTAN. Malgré ses directions de descente relativement mauvaises, la technique d'approximation quadratique a montré un bon comportement du point de vue temps de calcul, ceci étant dû au fait que toute recherche unidimensionnelle a été supprimée dans cet algorithme.

c) Réseau de Winnipeg

La meilleure borne inférieure obtenue au cours des essais pour la valeur optimale de la fonction économique est :

MBINF = 887 684,41

1. Frank & Wolfe

Pour un réseau de la taille de celui de Winnipeg, le temps passé dans la recherche unidimensionnelle devient une part importante du temps total (évaluations coûteuses de la fonction). Le paramètre ACCUR (cf. Annexe A) nécessite donc un ajustement plus soigneux que précédemment. Le tableau 13 donne la valeur atteinte par la fonction économique pour différents nombres d'itérations ainsi que le temps CPU (secondes) nécessaire pour y parvenir. On y remarque qu'une précision de 1 est nettement suffisante; mieux, c'est avec ACCUR=1 qu'on obtient le meilleur résultat en ce qui concerne le temps nécessaire pour atteindre un gap plus petit que 1%. Nous avons donc adopté cette valeur pour les autres algorithmes.

	précision dans la recherche unidimensionnelle (ACCUR)			
itérations	.001	.1	1.0	10.0
10	917,725.96 275.07	917,725.96 270.03	917,725.98 263.24	917,726.58 261.46
20	901,723.87 535.35	901,723.87 522.24	901,723.92 512.38	901,724.88 505.12
30	896,102.63 790.67	896,102.63 771.23	896,102.67 758.87	896,341.22 744.03
50	892,047.21 1,301.91	892,047.22 1,270.48	892,047.26 1,253.03	892,118.40 1,219.76
GAP < 1%	919.13	896.24	882.98	934.01
temps moy. pour une itér.	25.95	25.28	24.90	24.23

Tableau 13

2. Heuristique

Si cette méthode a été profitable sur l'algorithme de F &W dans les cas Dupuis et Hull, ce n'est plus le cas ici. Les résultats obtenus en prenant C=1,8 comme dans les cas précédents n'ont montré aucune amélioration sur F &W. Nous n'analyserons donc pas plus avant cette méthode dans le cas du réseau qui nous intéresse.

3. Partan

Nous avons testé ici les trois versions suivantes de PARTAN :

PARTAN 1 : Le pas PARTAN maximal est calculé avec la formule (23) et donc le domaine des valeurs admissibles pour le pas est $[0,p^{max}]$.

PARTAN 2 : p^{max} est calculé avec la formule (24) et on oblige le pas PARTAN à être plus grand que ou égal à 1; son domaine de variation est donc $[1,p^{max}]$.

PARTAN 3 : C'est la version réoptimisée. Comme nombre d'itérations comprises entre deux réoptimisations, nous avons repris la valeur optimale trouvée pour Hull, à savoir 4.

La figure 14 montre que les deux premières versions sont les meilleures,la version 2 fournissant d'un peu meilleures directions de descente au début de l'algorithme que la version 1. Nous retiendrons donc cette deuxième version pour les comparaisons.

4. Restriction et approximation quadratique

Nous n'avons pas refait dans ce cas d'études poun le choix des paramètres et les valeurs optimales trouvées précédemment ont été adoptées, à savoir :

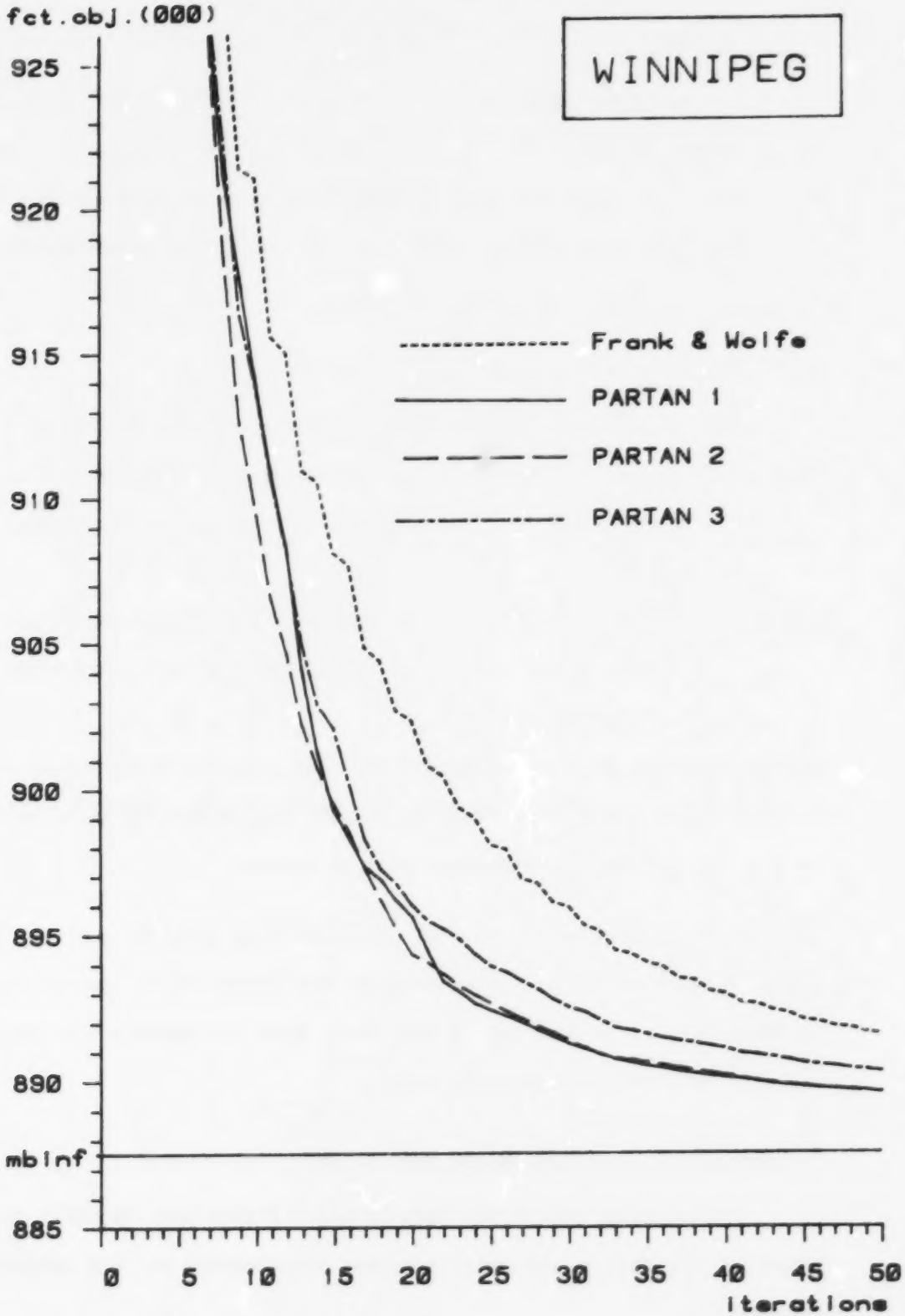
fct.obj.(000)
WINNIPEG
925
920
915
910
905
900
895
890
mbinf
885
Frank & Wolfe
PARTAN 1
PARTAN 2
PARTAN 3
0
5
10
15
20
25
30
35
40
45
50
iterations

Figure 14

restriction : NBITER=5, MAXPTS=6, IPARTA=1

approximation quadratique : NBITER=5, MAXIT=4+1(2), IPARTA=1.

5. Comparaisons

La figure 15 fournit l'évolution de la fonction économique en fonction du nombre d'itérations. A nouveau, on y constate que ce sont les méthodes PARTAN et restriction qui donnent les meilleures directions de descente, ces deux méthodes ne se distinguant pas entre elles.

En ce qui concerne l'efficacité au niveau du temps de calcul, comme observé auparavant, la technique d'approximation quadratique profite de son absence de recherche unidimensionnelle pour égaler les résultats obtenus par PARTAN au voisinage de l'optimum. PARTAN se révèle toutefois être la méthode la plus efficace pour des temps plus petits que mille secondes, c'est-à-dire pour les temps réellement utilisés dans la pratique.

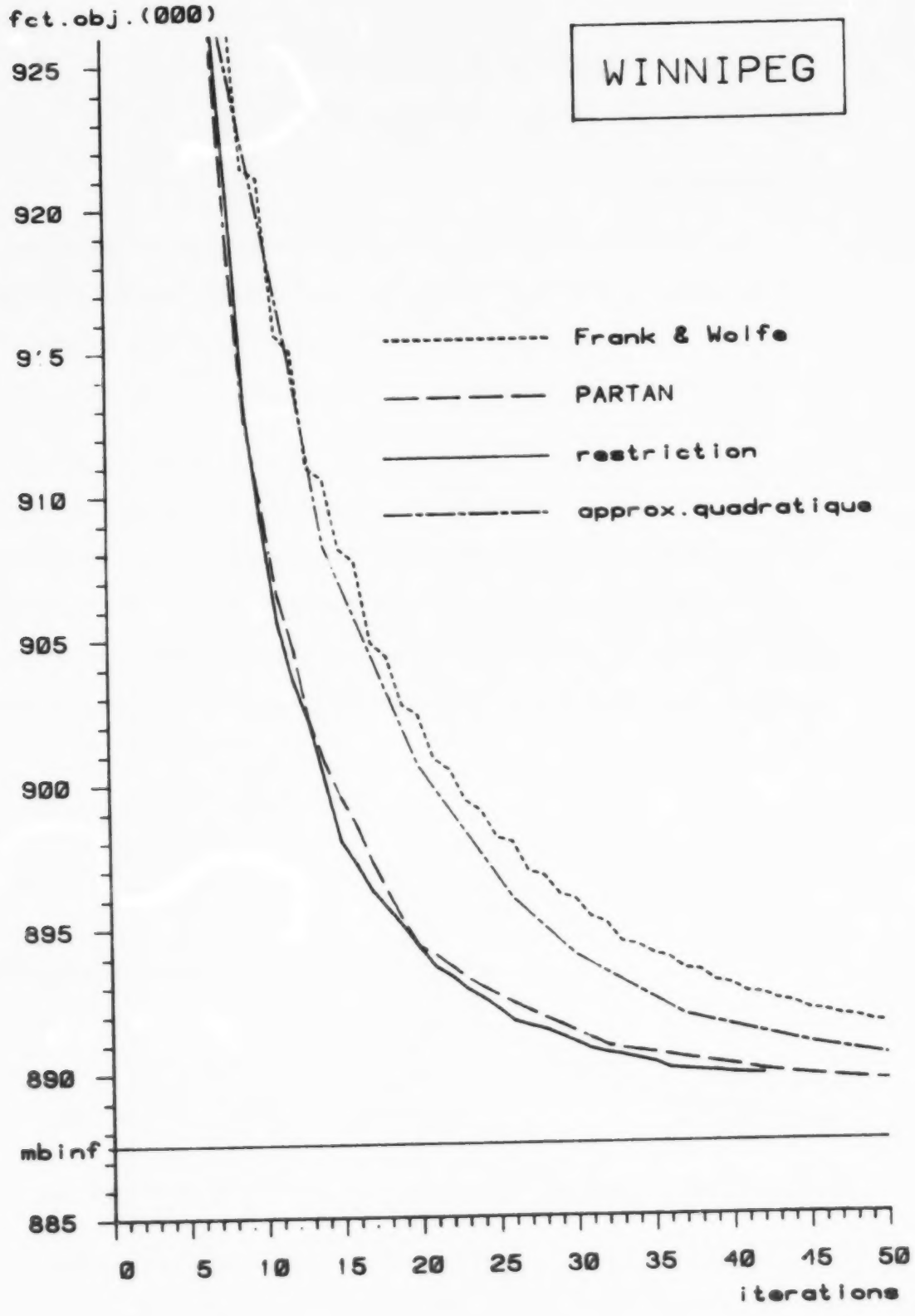
fct.obj.(000)
WINNIPEG
925
920
915
910
905
900
895
890
mbinf
885
0
5
10
15
20
25
30
35
40
45
50
iterations
Frank & Wolfe
PARTAN
restriction
approx.quadratique

Figure 15

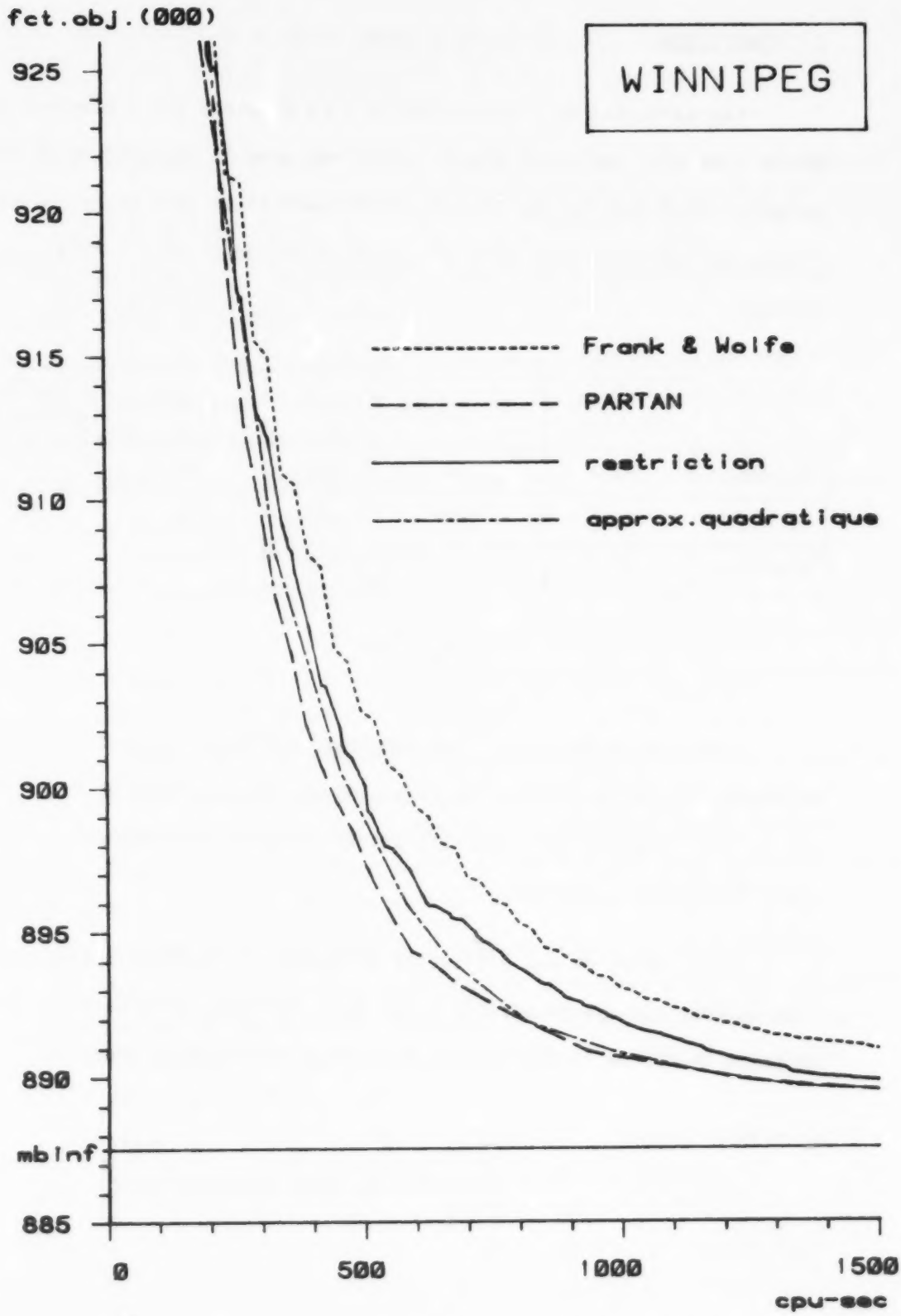
fct.obj.(000)
WINNIPEG
925
920
915
910
905
900
895
890
mbinf
885
0
500
1000
1500
cpu-sec
Frank & Wolfe
PARTAN
restriction
approx.quadratique

Figure 16

6. CONCLUSIONS

Les variantes de l'algorithme de F &W étudiées ici présentent toutes de sensibles améliorations en ce qui concerne la rapidité de la convergence, sauf dans le cas de la méthode heuristique pour grands réseaux. L'ordre des améliorations observées (% du temps de calcul) est le suivant :

	itérations de F & W	heuristique	PARTAN	restriction	approx. quadratique
Dupuis	93	57%	69%	62%	–
Hull	86	41%	58%	73%	70%
Winnipeg	60	–	40%	25%	38%

Tableau 14

Comme on le constate, les résultats diffèrent suivant la taille du réseau considéré. Si pour un réseau moyen, la stratégie de restriction est la plus intéressante, pour les petits et grands réseaux par contre, c'est PARTAN qui l'emporte.

Si du point de vue efficacité en temps CPU la méthode d'approximation quadratique est comparable à ces deux méthodes, elle ne l'est plus du point de vue de la qualité des directions de descente générées.

RÉFÉRENCES

[1] AASHTIANI, H., MAGNANTI, T., "Equilibria on a congested transportation network". SIAM Journal on Algebraic and Discrete Methods 2, 213-226 (1981).

[2] BAZARAA, M.S., SHETTY, C.M., "Nonlinear Programming. Theory and Algorithms". Wiley (1979).

[3] DAFERMOS, S.C., "The General Multimodal Network Equilibrium Problem with Elastic Demand". Lefschetz Center for Dynamical Systems, Brown University (1980).

[4] DEMBO, R.S., TOLOWITZKI, U., "Successive Inexact Quadratic Programming and its Application to Traffic Equilibrium Problems". School of Organization and Management, Yale University, Working Paper Series B #65 (1983).

[5] FLORIAN, M., "An Improved Linear Approximation Algorithm for the Network Equilibrium (Packet Switching) Problem". IEEE Proceedings, Decision and Control, 812-818 (1977).

[6] FLORIAN, M., "An Introduction to Network Models Used in Transportation Planning". Publication #262, Centre de recherche sur les transports, Université de Montréal. Présenté au "First Course on Transportation Planning Models of the International School of Transport Planning", Amalfi, Italie, 11-16 octobre 1982.

[7] FLORIAN, M., "Nonlinear Cost Network Models in Transportation Analysis". Publication #287, Centre de recherche sur les transports, Université de Montréal. Présenté à NETFLOW'83 - International Workshop on Network Flow Optimization Theory and Practice, Pise, Italie, 28-30 mars 1983.

[8] FRANK, M., WOLFE, P., "An algorithm for quadratic programming". Naval Research Logistic Quarterly 3, 95-110 (1956).

[9] GEOFFRION, A.M., "Elements of Large Scale Mathematical Programming", in Perspectives on Optimization, edited by A.M. Geoffrion, Addison-Wesley (1972).

[10] GILL, P.E., MURRAY, W., WRIGHT, M.H., "Practical Optimization". Academic Press (1981).

[11] HOLLOWAY, C.A., "An extension of the Frank and Wolfe method for feasible directions". Mathematical Programming 6, 14-27 (1974).

[12] LE BLANC, L.J., HELGASON, R.V., BOYCE, D.E., "Improved Efficiency of the Frank-Wolfe Algorithm". Working Paper #81-131, Owen Graduate School of Management, Vanderbilt University (1982).

[13] LUENBERGER, D.G., "Introduction to Linear and Nonlinear Programming". Addison-Wesley, Reading, Mass. (1973).

[14] NGUYEN, S., "A Unified Approach to Equilibrium Methods of Traffic Assignment". In Traffic Equilibrium Methods, Proceedings 1974, M. Florian, ed., Vol. 118, Lecture Notes in Economics and Mathematical Systems, Springer Verlag 148-182 (1976).

[15] NGUYEN, S., JAMES, L., "TRAFFIC, and Equilibrium Traffic Assignment Program". Publication #17, Centre de recherche sur les transports, Université de Montréal (1975).

[16] NGUYEN, S., DUPUIS, C., "Une méthode efficace de calcul d'un trafic d'équilibre dans les cas des coûts non-symétriques". Publication #205, Centre de recherche sur les transports, Université de Montréal (1981).

[17] SHAH, B.V., BUEHLER, R.J., KEMPTHORNE, O., "Some algorithms for minimizing a function of several variables". SIAM Journal of Applied Mathematics 12, 74-92 (1964).

[18] SMITH, M.J., "The existence, uniqueness and stability of traffic equilibria". Transportation Research 138, 295-304 (1979).

[19] SORENSON, H.W., "Comparison of some conjugate direction procedures for function minimization". Journal of the Franklin Institute 288, #6, 421-441 (1969).

[20] WARDROP, J., "Some Theoretical Aspect of Road Traffic Research". Proceedings of the Institute of Civil Engineers, Part II, Vol. 1, 325-378 (1952).

ANNEXE A

EFFICACITÉ DU PROGRAMME

L'intérêt du présent travail se situant surtout dans la comparaison de la rapidité de convergence de différents algorithmes, l'aspect d'efficacité du programme a été une préoccupation de second ordre. Le code de base est TRAFFIC (cf. Nguyen et James [15]). Outre les adaptations nécessaires à l'implantation des différents algorithmes étudiés, deux modifications majeures ont été apportées à ce code. La première concerne la recherche des plus courts chemins. Initialement, cette recherche s'effectuait avec l'algorithme de Dijkstra implanté avec un "heap". La version utilisée ici résoud ce problème avec l'algorithme de d'Esopo-Pape qui s'est révélé nettement plus efficace pour le type de réseau employé :

Réseau	Dupuis	Hull	Winnipeg
# d'itérations de F & W	50	30	25
temps CPU (sec.)			
Dijkstra	1,35	90	1310
d'Esopo-Pape	1,03	56	663
Gain	24%	38%	49%

Tableau A1

La seconde modification concerne la recherche unidimensionnelle. La problème correspondant est le suivant :

$$\min_{0\leq\lambda\leq\Delta} f(\lambda) = F(v^{\ell}+\lambda d^{\ell}) = \sum_{a\in A} S_a(v_a^{\ell}+\lambda d_a^{\ell}) \qquad (A1)$$

où : $S_a(x_a) = \int_0^{x_a} s_a(v)dv$

et $d_a^\ell = y_a^\ell - v_a^\ell$:

La méthode utilisée dans TRAFFIC pour résoudre ce problème est celle de la section dorée. Cette méthode possède un intervalle d'incertitude de longueur $(\frac{1}{\tau})^n \Delta$ où :

Δ = longueur de l'intervalle de recherche

n = nombre d'itérations de la méthode

$n+1$ = nombre d'évaluations de la fonction

τ = nombre d'or = $\frac{1+\sqrt{5}}{2}$ => $\frac{1}{\tau}$ = 0,618

$$f' = \frac{df}{d\lambda} = \sum_{a \in A} \frac{d}{d\lambda} \left[\int_0^{v_a^\ell + \lambda d_a^\ell} s_a(v)dv \right]$$

$$\text{ou : } f'(\lambda) = \sum_{a \in A} s_a(v_a^\ell + \lambda d_a^\ell) d_a^\ell \qquad \text{(A2)}$$

F étant convexe, f l'est aussi, donc f' est monotone croissante. La connaissance analytique de la dérivée permet de transformer le problème de minimisation (A1) en un problème de recherche de zéro d'une fonction monotone croissante. Parmi les méthodes pour résoudre ce nouveau problème, figurent la bissection et la sécante. La première fournit un intervalle d'incertitude de $(\frac{1}{2})^n \Delta$, mais peut converger lentement. La seconde possède un taux de convergence super-linéaire $r \approx 1,618$, mais peut diverger :

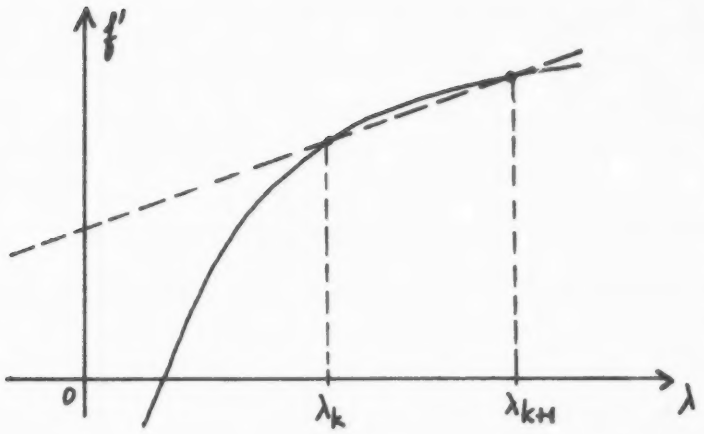

Figure A1

Pour remédier à cette situation, on modifie la méthode de la sécante de la façon suivante : λ_{k+1} remplace l'un parmi λ_k et λ_{k-1} (et non forcément λ_k) suivant le signe de la fonction : on veut toujours que la fonction ait des signes opposés aux bornes de l'intervalle d'incertitude. On obtient ainsi la méthode appelée "regula falsi". Il est à noter que cette modification peut détruire la convergence rapide de la méthode originale (cf. Gill et coll. [10]). Dans l'exemple donné à la figure A2, le point λ^0 ne sera jamais modifié et l'intervalle d'incertitude sera toujours plus grand que $(\lambda_0-\lambda^*)$.

L'idée est alors de combiner bissection et regula falsi.

L'algorithme finalement implanté a donc pris la forme :

0. (*initialisation*)

   $\lambda_A:=0$; $\lambda_B:=\Delta$ (*[λ_A,λ_B] = intervalle d'incertitude*)

   Si $f'(\lambda_A)$ et $f'(\lambda_B)$ ont même signe (*c'est-à-dire, comme f' est monotone croissante, f' ne s'annule pas dans [0,Δ], donc le min se

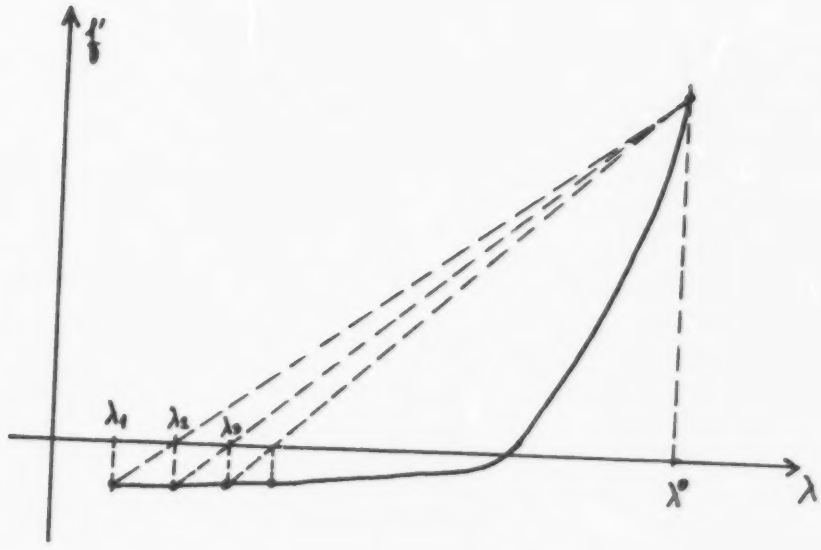

Figure A2

trouve en λ_A ou en λ_B*)

alors si $f'(\lambda_A) \leq f'(\lambda_B)$ alors $\lambda_{opt} := \lambda_A$

sinon $\lambda_{opt} := \lambda_B$

STOP

1. (*bissection*)

$\lambda := \frac{1}{2}(\lambda_A + \lambda_B)$

Si $f'(\lambda_A)$ et $f'(\lambda)$ ont même signe alors $\lambda_A := \lambda$

sinon $\lambda_B := \lambda$

2. (*regula falsi*)

$$\lambda := \lambda_B - \frac{\lambda_B - \lambda_A}{f'(\lambda_B) - f'(\lambda_A)} \cdot f'(\lambda_B)$$

critère d'arrêt : Si $|f'(\lambda)| <$ ACCUR alors $\lambda_{opt} :=$, STOP

Si $f'(\lambda_A)$ et $f'(\lambda)$ ont même signe alors $\lambda_A := \lambda$

sinon $\lambda_B := \lambda$

Retour à 1.

Le gain en temps de calcul réalisé en introduisant cette nouvelle routine de recherche linéaire s'est révélé être de 30% à 50%.

ANNEXE B

LE PROGRAMME UTILISÉ

Comme signalé, c'est une adaptation de TRAFFIC [15] qui a été utilisée.

Les sous-routines suivantes ont été ajoutées :

- REAL FUNCTION FPRIM

Cette fonction, utilisée dans la recherche unidimensionnelle, calcule la dérivée directionnelle de la fonction économique à l'aide de la formule (A2).

- SUBROUTINE LS

Sous-routine remplaçant la recherche unidimensionnelle suivant la section d'or, décrite à l'Annexe A.

- SUBROUTINE SUMS

Calcule $WEWSUM = \nabla F(v^{\ell})(y^{\ell}-v^{\ell})$

$$= \sum_{a \in A} s_a(v_a^{\ell}) d_a^{\ell}$$

- SUBROUTINE EXFLOW

Fournit un nouveau point extrémal.

- SUBROUTINE UPDTGT et UPDATG

Effectue l'ajustement des poids α_i à l'étape 3 de l'algorithme avec stratégie de restriction.

- SUBROUTINE NEWFLO

Fournit la direction de descente dans la restriction par la méthode du gradient réduit.

Si la dimension des réseaux Dupuis et Hull ne pose pas de problème quant à l'espace mémoire requis, ce n'est pas le cas pour celui de Winnipeg. L'utilisation de tableaux pour plusieurs items différents est alors nécessaire. C'est le code de cette version que nous fournissons ci-après. Ce code permet d'utiliser l'algorithme de Frank & Wolfe ainsi que les deux variantes PARTAN (avec formule (23)) et stratégie de restriction. Pour obtenir la variante heuristique, la modification à apporter est triviale. Pour la variante quadratique, l'algorithme fourni en page 39 indique de façon évidente les adaptations à apporter.

```
      PROGRAM TRAFFIC(INPUT,OUTPUT,TAPE2,TAPE3,TAPE8,TAPE9,TAPE10,
     +                TAPE11,TAPE6=OUTPUT)
C
C--------THIS PROGRAM PERFORMS THE ASSIGNMENT OF TRAFFIC TO LINKS IN A
C        GIVEN TRANSPORTATION NETWORK TO SATISFY THE DEMAND BETWEEN THE
C        ORIGIN/DESTINATION PAIRS IN THE NETWORK SUCH THAT THE COST TO
C        EACH INDIVIDUAL USER F THE NETWORK IS MINIMIZED.
C
C--------THE PROGRAM IS BASED ON THE ALGORITHM DESCRIBED IN
C          "A MATHEMATICAL PROGRAMMING APPROACH TO EQUILIBRIUM METHODS OF
C          TRAFFIC ASSIGNMENT WITH FIXED DEMANDS" BY SANG NGUYEN
C          (UNIVERSITE DE MONTREAL, DEPT DE INFORMATIQUE,
C          TECHNICAL REPORT, NUMBER 138, AUGUST 1973), AND
C       MODIFIED TO INCLUDE RESTRICTION AND SOLUTION WITH
C       REDUCED GRADIENT BY M. FLORIAN.
C       MODIFIED TO INCLUDE PARTAN TECHNIQUE BY J. GUELAT.
C
       COMMON/OMEGAB/OMEGA(31),DN(31),H(31)
       COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
     +               ND(2840),ND(2840),NULACO(2840),DIST(2840)
       COMMON /NET2/ VCUM(2840),TRAV(2840), FLOW(2840),ODIJ(150)
       COMMON        WKMEM(3300)
       COMMON /PARAM/ EPSI,TIMEMA,MAXCYC,MAXIT,IPCH,IPRT,IPOST,IFACS
       COMMON /STAT/ RUNSEC(10), TOTDEM
       COMMON /COEF/ ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
       COMMON /EDIT/ TITLE(12),RLINE(10),LINE(9),L1(26),L2(26)
       COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL,KEPFIL,MEMFIL
       COMMON /PARM2/ ACCUR,MAXPTS,NBITER,IPARTA
       DIMENSION OMEGA1(100),OMEGA2(100),AUX(100)
       INTEGER CARD,ODFIL,NETFIL,SOLFIL,RESUL
       REAL LAMBDA,ISLSEC
       EQUIVALENCE (NORIG,LINE(1)),(NDEST,LINE(2)),(IDIST,LINE(3)),
     +             (NULA,LINE(4)),(DEFUNC,LINE(5)),(ISPEED,LINE(6)),
     +             (ITOBS,LINE(7)),(IVOBS,LINE(8)),(LTYPE,LINE(9))
       EQUIVALENCE (RDSEC,RUNSEC(1)),(ISLSEC,RUNSEC(2)),
     +             (PRTSEC,RUNSEC(3)),(OSMSEC,RUNSEC(4)),
     +             (PTHSEC,RUNSEC(5))
C
       DATA MXRUNS/5/
       DATA OMEGA1/100*0.0/
       DATA OMEGA2/100*0.0/
C
C
       ODFIL=2
       CARD=9
       SOLFIL=3
       KEPFIL=11
       NETFIL=8
       MEMFIL=10
 1002  FORMAT(/"0I01 RUN TERMINATED.....TIME LIMIT :",E15.7)
```

```
 1003 FORMAT(/'0102 RUN TERMINATED      CYCLE LIMIT   ',I8)
 1004 FORMAT(/'0103 RUN TERMINATED      ITERATION LIMIT   ',I5)
 1013 FORMAT ('0 EPSILON =',E15.7/'  TIMEMA  =',E15.7/
     +       '  MAXCYC  =',I8)
 1014 FORMAT ('0TOTAL VEHICLE HOURS =',F12.2/
     +        '0TOTAL VEHICLE MILES =',F12.2/
     +        '0AVERAGE SPEED       =',F12.2//)
 1019 FORMAT ('0 CPU SECS FOR - NETWORK DATA INPUT',T48,'=',F10.2/
     +        T16,'- SET UP INITIAL SOLUTION',T48,'=',F10.2/
     +        T16,'- RE-ESTABLISH PARTIAL SOLUTION',T48,'=',F10.2/
     +        T16,'- LINE SEARCH (TOTAL)',T48,'=',F10.2/
     +        T16,'- FINDING SHORTEST PATH (TOTAL)',T48,'=',F10.2)
 2000 FORMAT (1H1,100X,'PAGE ',I4//30X,5A8/18X,7A8//
     +        4X,'LINK    NODE      NODE      LENGTH     NUMB    DELAY',
     +        4X,'OBSERVED   OBSERVED    OBSERVED    LINK',
     +        2X,'PREDICTED  PREDICTED   PARAMETERS'/
     +        4X,'IDEN    FROM      TO',14X,'LANES   FUNCT     TRAVEL',
     +        6X,'TRAVEL      VOLUME      TYPE   TRAVEL     VOLUME',
     +        8X,'A',6X,'B'/
     +        55X,'SPEED',8X,'TIME',26X,'TIME'/
     +        29X,'MILES',21X,'(MPH)',7X,'(MINS)',24X,'(MINS)'//)
 2003 FORMAT (1H1,30X,5A8/18X,7A8//)
 2005 FORMAT (I4,3(I4,4X),F6.2,4X,I3,5X,I2,8X,F4.1,6X,F6.2,4X,I8,
     +        7X,I2,3X,F6.2,F12.4,5X,I4,3X,I4)
 2004 FORMAT(5X,3(I4,1X),I3,I2,1X,I4,1X,2(I5,1X),I3,1X,I4,1X,7I4)
 2006 FORMAT (I4,3(I4,4X),F6.2,4X,I3,5X,I2,8X,F4.1,6X,F6.2,4X,A8,
     +        7X,I2,3X,F6.2,F12.4,5X,I4,3X,I4)
 2011 FORMAT(10F8.0)
 2012 FORMAT (10F8.2)
 2013 FORMAT (20I4)
 3000 FORMAT (//1X,'IOB END CYCLE',I4,'    ITERATION',I4,
     +       '  OBJ FUNCTION = ',E16.10)
 3001 FORMAT (/21X,'BEST BOUND = ',E16.10,'    GAP1 = ',F6.2,' % ',
     +        '  GAP2 = ',F6.2,' % ','  CUM TIME = ',F7.2)
 3002 FORMAT (/21X,'COUT MOYEN AU POINT COURANT  ',F7.2,' MIN ',
     +        '  AU POINT EXTR.  ',F7.2,' MIN ')
 3003 FORMAT('1I09 INITIAL SOLUTION-VALUE OF OBJ  FUNCTION',E20.10)
 3004 FORMAT(2X,'I10 DEL(F) < EPSILON ',E20.10)
 4001 FORMAT(10A8)
C
C-----INITIALISATION
C
      CALL SECOND (TBEGIN)
      REWIND MEMFIL
      TPARTA = 0.0
      TNEW=0.0
      WBEND=0.0
      REWIND KEPFIL
      IT=0
       MAXKEP =30
```

```
1010=       OMEGA(1)=1.0
1020=       DO 50 I=1,10
1030=    50 RUNSEC(I)=0.
1040=       ITERAT = 0
1050=C
1060=C-----READ NETWORK DATA
1070=C
1080=       CALL :ETWOR
1090=       CALL SECOND(TTOUT)
1100=       RDSEC=TTOUT-TBEGIN
1110=       TTIN =TTOUT
1120=C
1130=C
1140=C********************************** BEGINNING OF THE ALGORITHM
1150=C
1160=C      I----------I
1170=C      I  STEP 0  I
1180=C      I----------I
1190=C
1200=C-----CHECK FOR POST OPTIMIZATION
1210=C
1220=       IF(IPOST.LT.1)GOTO 60
1230=           CALL PARTSO
1240=           CALL SECOND(TTOUT)
1250=           PRTSEC = TTOUT - TTIN
1260=           GO TO 70
1270=C
1280=C-----SET UP INITIAL (ALL OR NOTHING) SOLUTION
1290=C
1300=    60 REWIND ODFIL
1310=       DO 40 I=1,NLINK
1320=           VOLM(I)=0.0
1330=           FLOW(I)=0.0
1340=    40     TRAV(I)=USERCST(I)
1350=C
1360=       CALL EXFLOW
1370=       ITERAT = ITERAT + 1
1380=C
1390=       DO 43 I=1,NLINK
1400=    43     VOLM(I)=FLOW(I)
1410=       WRITE (MEMFIL) FLOW
1420=       DO 44 I=1,NLINK
1430=    44     TRAV(I) = USERCST(I)
1440=       CALL SECOND(TTOUT)
1450=       IBLSEC=TTOUT-TTIN
1460=C
1470=    70 CONTINUE
1480=       TTIN=TTOUT
1490=C
1500=C-----CALCULATE OBJECTIVE FUNCTION FOR CURRENT SOLUTION
```

```
1510=C
1520=C-----CONST = CONSTANT TERM OF OBJECTIVE FUNCTION
1530=C           = TOTAL COST, GIVEN USERCS  VALUES CALCULATED FOR ZERO
1540=C             VOLUMES
1550=C-----SUMTRA (0.0,0) = VARIABLE TERM OF OBJECTIVE FUNCTION
1560=C                      = TOTAL COST, GIVEN USERCS  VALUES CALCULATED
1570=C                        USING INITIAL SOLUTION VOLUMES.
1580=C                                        (LAMBDA = 0.0)
1590=       CONST=0.0
1600=C
1610=       OBF=SUMTRA(0.0,0) + CONST
1620=C
1630=       PRINT 3003, OBF
1640=C
1650=C-----INITIALISATIONS
1660=C
1670=       U=1
1680=       VAL=OBF
1690=       LIMKEP=1
1700=       NUMCYC=1
1710=       DO 45 I=1,NLINK
1720=    45 TRAV(I) = USERCST(I)
1730=C
1740=C
1750=C      I----------I
1760=C      I  STEP 1  I
1770=C      I----------I
1780=C
1790=10000 NUMCYC=NUMCYC+1
1800=C
1810=       CALL EXFLOW
1820=       ITERAT = ITERAT + 1
1830=C
1840=       CALL SUMS(NEWSUM, GRADX, GRADY)
1850=C
1860=       CALL SECOND(TTOUT)
1870=C
1880=       WWBND1=OBF+NEWSUM
1890=       IF(WWBND1.GT.WWBND) WWBND=WWBND1
1900=C
1910=C-----STOPPING CRITERION
1920=C
1930=       IF(ABS((OBF-WWBND)/OBF).LT.EPSI) GOTO 300
1940=C
1950=C      GO TO STEP 3
1960=       GOTO 10100
1970=C
1980=C
1990=C      I----------I
2000=C      I  STEP 2  I      *** RESTRICTION ***
```

```
C          I----------I
C
10050 CALL SECOND(TN1)
C
          CALL NEWFLO (WEWSUM, LIMKEP )
C
          CALL SECOND(TN2)
          TNEW=TNEW+TN2-TN1
C
C-----COST AT VCUM
C
          GRADX = 0.0
          DO 10051 J=1, NLINK
10051        GRADX = GRADX + VCUM(J)*TRAV(J)
C
          IT=IT+1
          NUMCYC=NUMCYC+1
          IF(NUMCYC.GT.MAXCYC) GOTO 211
C
C-----STOPPING CRITERION OF RESTRICTION
C
          IF(ABS((WEWSUM)/OBF).LE.0.001.OR. IT.GE.5) GOTO 10200
C
C
C          I--- ------I
C          I  STEP 3  I       *** LINE SEARCH ***
C          I----------I
C
10100 CALL SECOND(TTIN)
C
          CALL LS(LAMBDA, 1.00, ACCUR, VAL, 0)
C
          PRINT 10110, LAMBDA
10110 FORMAT(/21X, "PAS FRANK & WOLFE :  ", F10.5/)
C
          CALL SECOND(TTOUT)
          GSMSEC = GSMSEC +TTOUT-TTIN
C
          DO 10101 J=1, NLINK
          X=VCUM(J)+LAMBDA*FLOW(J)
          VCUM(J)=X
          TRAV(J)=USERCST (J)
10101 CONTINUE
C
          OBF=VAL + CONST
C
          CALL SECOND(TTOUT)
          TTOUT=TTOUT-TBEGIN
          XYZ = 100*ABS(WEWSUM/OBF)
          GAP = 100*ABS((OBF-WWBND)/OBF)
```

```
2510=       SPEED1 = GRADX/TOTDEM
2520=       SPEED2 = GRADY/TOTDEM
2530=       PRINT 3000, NUMCYC, ITERAT, OBF
2540=       PRINT 3001, WWBND, XYZ, GAP, TTOUT
2550=       PRINT 3002, SPEED1, SPEED2
2560=C
2570=C-----CHECKING TIME LIMIT
2580=C
2590=       IF(TTOUT.LT.TIMEMA ) GOTO 10102
2600=       PRINT 1002, TIMEMA
2610=       GOTO 300
2620=C
2630=C-----CHECK THE NUMBER OF ALLOWABLE CYCLES
2640=C
2650=10102 IF (NUMCYC.LT.MAXCYC) GOTO 10107
2660=       PRINT 1003, MAXCYC
2670=       GOTO 300
2680=C
2690=C-----CHECK THE NUMBER OF ALLOWABLE ITERATIONS
2700=C
2710=10107 IF (ITERAT.LT.MAXIT) GOTO 10108
2720=       PRINT 1004, MAXIT
2730=       GOTO 300
2740=C
2750=C-----CHECKING FOR RESTRICTION
2760=C
2770=10108 IF (NUMCYC.LT.NBITER) GOTO 10106
2780=          IF (LIMKEP.EQ.1) GOTO 10222
2790=C
2800=C-----ADJUSTING THE OMEGAS
2810=C
2820=       IF(U.EQ.2) GOTO 10103
2830=       CALL UPDATO(LIMKEP ,LAMBDA)
2840=       GOTO 10105
2850=10103 CALL UPDTOT (LAMBDA, LIMKEP )
2860=10105 CONTINUE
2870=C
2880=          U=2
2890=C         GO TO STEP 2
2900=          GOTO 10050
2910=C
2920=10106 IF (IPARTA.EQ.0) GOTO 10000
2930=       IF (NUMCYC.GE.3) GOTO 10300
2940=          OMEGA1(1) = 1.0 - LAMBDA
2950=          OMEGA1(2) = LAMBDA
2960=          OMEGA2(1) = 1.0
2970=          WRITE (MEMFIL) VCUM
2980=          GOTO 10000
2990=C         TO STEP 1
3000=C
```

```
3010=C
3020=C       I----------I
3030=C       I  STEP 4  I      *** PARTAN LINE SEARCH ***
3040=C       I----------I
3050=C
3060=10300 CALL SECOND (TN1)
3070=C
3080=C-----SEARCH FOR THE MAXIMUM PARTAN STEP
3090=C
3100=        PMAX = 1.0E10
3110=        IAUX = NUMCYC - 2
3120=        DO 10301 J=1,IAUX
3130=           DEN = OMEGA2(J) - (1.0 - LAMBDA)*OMEGA1(J)
3140=           IF (DEN.LE.0.0) GOTO 10301
3150=           X = OMEGA2(J)/DEN
3160=           IF (X.LT.PMAX) PMAX = X
3170=10301 CONTINUE
3180=C
3190=        PRINT 9898,PMAX
3200= 9898 FORMAT(/21X,"PAS PARTAN MAX :  ",F10.5)
3210=C
3220=C-----PARTAN DIRECTION SEARCH
3230=C
3240=        REWIND MEMFIL
3250=        READ (MEMFIL) TRAV
3260=        DO 10302 J=1,NLINK
3270=10302 FLOW(J) = VCUM(J) - TRAV(J)
3280=C
3290=C-----PARTAN LINE SEARCH
3300=C
3310=        CALL LS (PAS,PMAX,ACCUR,VAL,1)
3320=C
3330=        PRINT 8989,PAS
3340= 8989 FORMAT(21X,"PAS PARTAN :  ",F10.5)
3350=C
3360=        CALL SECOND (TN2)
3370=        TPARTA = TPARTA + TN2 - TN1
3380=C
3390=        DO 10303 J=1,NLINK
3400=10303      VCUM(J) = TRAV(J) + PAS*FLOW(J)
3410=        READ (MEMFIL) TRAV
3420=        REWIND MEMFIL
3430=        WRITE (MEMFIL) TRAV
3440=        WRITE (MEMFIL) VCUM
3450=        DO 10313 J=1,NLINK
3460=10313      TRAV(J) = USERCST(J)
3470=C
3480=C-----ADJUSTING THE OMEGA-IS
3490=C
3500=        IAUX = NUMCYC - 1
```

```
3510=       DO 10304 J=1,IAUX
3520=10304 AUX(J) = PAS*(1.0-LAMBDA)*OMEGA1(J) + (1.0-PAS)*OMEGA2(J)
3530=       AUX(NUMCYC) = PAS*LAMBDA
3540=C
3550=       DO 10305 J=1,NUMCYC
3560=          OMEGA2(J) = OMEGA1(J)
3570=          OMEGA1(J) = AUX(J)
3580=10305 CONTINUE
3590=C
3600=       GOTO 10000
3610=C      TO STEP 1
3620=C
3630=C
3640=C      I----------I
3650=C      I  STEP 5  I      *** NEW EXTREMAL FLOW ***
3660=C      I----------I
3670=C
3680=10200 IT=0
3690=       U=1
3700=       LIMKEP =LIMKEP +1
3710=       IF (LIMKEP .LT. MAXKEP) GOTO 10211
3720=          PRINT 1449,MAXKEP
3730= 1449     FORMAT (" ESPACE MEMOIRE DEMANDE INSUFFISANT",
3740=      +            " POUR LE TABLEAU IX",I4)
3750=          GOTO 300
3760=C
3770=10211 IF (LIMKEP.LE.MAXPTS) GOTO 10201
3780=10222     LIMKEP = 2
3790=          REWIND KEPFIL
3800=          WRITE (KEPFIL) VCUM
3810=          OMEGA(1) = 1.0
3820=          DO 10212  I=2,MAXPTS
3830=10212     OMEGA(I) = 0.0
3840=          PRINT 9875
3850= 9875     FORMAT(/" ON LAISSE TOMBER LES PTS EXTR. GENERES JUSQU'ICI")
3860=          GOTO 10213
3870=10201 PRINT 9876,NUMCYC,LIMKEP
3880= 9876 FORMAT(/" AJOUTE UN POINT EXTREMAL, CYCLE : ",I4,
3890=      +       " LIMKEP : ",I4)
3900=C
3910=10213 CALL EXFLOW
3920=       ITERAT = ITERAT + 1
3930=       WRITE(KEPFIL)FLOW
3940=C
3950=       CALL SUMS(WEWSUM,GRADX,GRADY)
3960=C
3970=       VAL=SUMTRA (0.0,0)
3980=       OBF=VAL+ CONST
3990=C
4000=       WWBND1=OBF+WEWSUM
```

```
4010=       IF(WWBND1.GT.WWBND) WWBND=WWBND1
4020=C
4030=C-----STOPPING CRITERION
4040=C
4050=       IF(ABS((OBF-WWBND)/OBF).GT.EPSI) GOTO 10202
4060=          PRINT 3004,EPSI
4070=          GOTO 300
4080=C
4090=C-----CHECK THE NUMBER OF ALLOWABLE CYCLES
4100=C
4110=10202 IF(NUMCYC.LT.MAXCYC)GOTO 10203
4120=   211    PRINT 1003,MAXCYC
4130=          GOTO 300
4140=C
4150=10203 NUMCYC=NUMCYC+1
4160=C
4170=C-----CHECK THE RUNNING TIME
4180=C
4190=       IF(TTOUT.LT.TIMEMA )GOTO 10204
4200=          PRINT 1002,TIMEMA
4210=          GOTO 300
4220=C
4230=10204 CONTINUE
4240=C      GO TO STEP 3
4250=       GOTO 10100
4260=C
4270=C
4280=C********************************** END OF THE ALGORITHM
4290=C
4300=C
4310=   300 CONTINUE
4320=       PRINT 2003,TITLE
4330=       PRINT 1013,EPSI,TIMEMA ,MAXCYC
4340=       PRINT 1019,(RUNSEC(I),I=1,MXRUNS)
4350=       PRINT 1018,TNEW
4360= 1018 FORMAT(T16,"- REDUCED GRAD TIME (TOTAL)",T48,"=",F10.2)
4370=       PRINT 1017,TPARTA
4380= 1017 FORMAT(T16,"- PARTAN SEARCH (TOTAL) ",T48,"=",F10.2//)
4390=C
4400=C--------CALCULATE TOTAL VEHICLE HOURS AND MILES
4410=       VEHRS = 0.0
4420=       VEMLS = 0.0
4430=       DO 360 I=1,NLINK
4440=   352 VEHRS = VEHRS + VCUM(I) * TRAV(I)
4450=       VEMLS = VEMLS + VCUM(I) * DIST(I)
4460=   360 CONTINUE
4470=       VEHRS = VEHRS / 60.
4480=       VEMLS = VEMLS / 100
4490=       IF(VEHRS.LT.1.0E-9)GO TO 318
4500=       VESPD = VEMLS / VEHRS
```

```
C
  318 PRINT 1014, VEHRS, VEMLS, VESPD
C
      IF(IPRT.LE.0)GOTO 100
C
C------PRINT OUT THE CALCULATED LINK FLOWS AND TRAVEL TIMES
C
  317 REWIND NETFIL
      REWIND SOLFIL
 5002 FORMAT(10F8.0)
      WRITE(SOLFIL,4001) RLINE
      NLINE = 45
      NPAGE = 1
      LKXXX=NLINK
      NA=0
      DO 90 N = 1,NLINK
C--------PRINT PAGE HEADING EVERY 45 LINES
      IF (NLINE .LT. 45) GO TO 82
   80     PRINT 2000, NPAGE, TITLE
          NLINE = 0
          NPAGE = NPAGE + 1
C
   82 NLINE = NLINE + 1
      READ (NETFIL,2004) LINE
      ADIST = IDIST / 100.0
      SPEED = ISPEED / 10.0
      TOBS = ITOBS / 100.0
      TPRED=TRAV(N)
      IF (IVOBS .GT. 0) GO TO 85
C         IVOBS = 8H   NA
      NA=8H   NA
          PRINT 2006, N, LKXXX, NORIG, NDEST, ADIST, NULA, DEFUNC, SPEED, TOBS,
     +          NA, LTYPE, TPRED, VCUM(N), NPARA, NPARB
          GO TO 90
   85 PRINT 2005, N, LKXXX, NORIG, NDEST, ADIST, NULA, DEFUNC, SPEED, TOBS,
     +                IVOBS, LTYPE, TPRED, VCUM(N), NPARA, NPARB
   90 CONTINUE
  100 WRITE(SOLFIL,2011) (VCUM(I), I=1,NLINK)
      WRITE(SOLFIL,2012) (TRAV(I), I=1,NLINK)
      WRITE (SOLFIL,2013) (KD(I), I=1,NCENT)
      REWIND SOLFIL
 5001 FORMAT(10(I5,F8.0))
 5000 FORMAT(10F8.0)
      STOP
      END
C
C*****************************************************************
      SUBROUTINE  MINTRE (NN)
C
C--------CONSTRUCT SHORTEST PATH TO ALL DESTINATIONS FORM ORIGIN NN
```

```
C
      COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
     +              ND(2840),ND(2840),NULACO(2840),DIST(2840)
      COMMON /NET2/ VCUM(2840),TRAV(2840), FLOW(2840),ODIJ(150)
      COMMON          WKMEM(3300)
      COMMON /STAT/ RUNSEC(10), TOTDEM
      DIMENSION LPR(1100),TSUM(1100),NSET(1100)
      EQUIVALENCE (TSUM(1),WKMEM(1)),(NSET(1),WKMEM(1101))
      EQUIVALENCE (LPR(1),WKMEM(2201))
      EQUIVALENCE (RDSEC,RUNSEC(1)),(ISLSEC,RUNSEC(2)),
     +            (PRTSEC,RUNSEC(3)),(QSMSEC,RUNSEC(4)),
     +            (PTHSEC,RUNSEC(5))
C
C
C
      CALL SECOND(TTIN)
C
C--------INITIALISATION
C
      MAX=999999
      DO 10 I=1,NNODE
      LPR(I)=0
      TSUM(I)= 1.0E10
      NSET(I)=0
      IF(NOLAB(I).EQ.0)NSET(I)=MAX
   10 CONTINUE
C
C--------NO CENTROID WILL BE SCANNED BUT THE ORIGIN
C
      DO 20 I=1,NCENT
   20 NSET(I)=MAX
C
C
      LAST=NN
      N=NN
      TSUM(N)=0.0
      GOTO 110
C
C--------FETCH NEW NODE FOR SCANNING.  IF LIST EMPTY STOP
C
  100 I=N
      N=NSET(N)
      IF(N.EQ.MAX)GOTO 200
      NSET(I)=-1
C
C-------SCAN NODE N
C
  110 I1=NOLAB(N)
      K=N
```

```
      120 K=K+1
          I2=NOLAB(K) -1
          IF(I2 LT. 0)GOTO 120
C
          DO 130 I=I1, I2
          M=ND(I)
          VAL=TSUM(N)+TRAV(I)
          IF(VAL .GE. TSUM(M))GOTO 130
          TSUM(M)=VAL
          LPR(M)=I
C
C---------ADD M TO LIST IF FIRST TIME REACH  M  OR LIST HAS
C---------ONLY ONE ELEMENT,  ELSE INSERT M AT TOP LIST IF
C---------M IS NOT ALREADY IN LIST
C
          IF(NSET(M))121, 122, 130
      121 IF(NSET(N) .EQ. MAX)GOTO 122
          NSET(M)=NSET(N)
          NSET(N)=M
          GOTO 130
      122 NSET(M)=MAX
          NSET(LAST)=M
          LAST=M
      130 CONTINUE
C
          GOTO 100
      200 CONTINUE
          CALL SECOND (TTOUT)
          PTHSEC = PTHSEC + TTOUT - TTIN
          RETURN
          END
C
C*****************************************************************************
C
          REAL FUNCTION SUMTRA (LAMBDA, MODE)
C
C---------CALCULATE OBJECTIVE FUNCTION (SUM TRAVEL TIME) FOR A
C         GIVEN LAMBDA.
C
          COMMON /NET1/ NNODE, NLINK, NCENT, KD(150), NOLAB(1100),
         +              NO(2840), ND(2840), NULACO(2840), DIST(2840)
          COMMON /NET2/ VCUM(2840), TRAV(2840),  FLOW(2840), ODIJ(150)
          COMMON /UNIT/ CARD, ODFIL, NETFIL, SOLFIL, KEPFIL, MEMFIL
          COMMON/COEF/  ALPHA(20), BETA(20), GAMMA(20), DELTA(20)
          INTEGER CARD, ODFIL, NETFIL, SOLFIL
          REAL LAMBDA
C
C
          SUM=0.0
          REWIND MEMFIL
          READ (MEMFIL) TRAV
          DO 100 J=1, NLINK
```

```
      VOL = (1-MODE)*VCUM(J) + MODE*TRAV(J) + LAMBDA*FLOW(J)
C---------UNPACK DELAY CURVE NUMBER  JC
      JC=MOD(NULACO(J),1000)
      IF (VOL.GE.0.0) GOTO 10
      VOL = 0.0
   10 CONTINUE
      IF (GAMMA(JC).EQ.0.0) GOTO 100
      NULA=NULACO(J)/1000
C---------IT IS ASSUMED THAT VOL IS EQUALLY DISTRIBUTED OVER LANES
      IF (ALPHA(JC).EQ.0.0) GOTO 2
      DELT=DELTA(JC)*FLOAT(NULA)/10.0
      IF (DELT.EQ.0.0) GOTO 100
      A2=(ALPHA(JC)/(BETA(JC)+1.0))*(VOL/DELT)**BETA(JC)
      SUM=SUM + (DIST(J)*.6/GAMMA(JC))*(1.0+A2)*VOL
      GOTO 100
    2 SUM = SUM + (DIST(J)*.6/GAMMA(JC))*VOL
  100 CONTINUE
      SUMTRA =SUM
      RETURN
      END
C
C*****************************************************************************
C
      REAL FUNCTION FPRIM (LAMBDA, MODE)
C
C          CALCULATE THE DERIVATIVE OF THE OBJECTIVE
C          FUNCTION FOR A GIVEN LAMBDA.
C
      COMMON /NET1/ NNODE, NLINK, NCENT, KD(150), NOLAB(1100),
     +              NO(2840), ND(2840), NULACO(2840), DIST(2840)
      COMMON /NET2/ VCUM(2840), TRAV(2840),  FLOW(2840), ODIJ(150)
      COMMON/COEF/  ALPHA(20), BETA(20), GAMMA(20), DELTA(20)
      COMMON /UNIT/ CARD, ODFIL, NETFIL, SOLFIL, KEPFIL, MEMFIL
      INTEGER CARD, ODFIL, NETFIL, SOLFIL
      REAL LAMBDA
C
C
      SUM = 0.0
      REWIND MEMFIL
      READ (MEMFIL) TRAV
      DO 100 J=1, NLINK
         VOL = (1-MODE)*VCUM(J) + MODE*TRAV(J) + LAMBDA*FLOW(J)
C--------UNPACK DELAY CURVE NUMBER JC
         JC = MOD(NULACO(J),1000)
         IF (VOL.GE.0.0) GOTO 10
         VOL = 0.0
   10    CONTINUE
         IF (GAMMA(JC).EQ.0.0) GOTO 100
         IF (ALPHA(JC).EQ.0.0) GOTO 2
         NULA = NULACO(J)/1000
```

```
6510=C--------IT IS ASSUMED THAT VOL IS EQUALLY DISTRIBUTED OVER LANE
6520=          DELT = DELTA(JC)*FLOAT(NULA)/10.0
6530=          IF (DELT.EQ.0.0) GOTO 100
6540=          A2 = ALPHA(JC)*(VOL/DELT)**BETA(JC)
6550=          SUM = SUM + (.6*DIST(J)/GAMMA(JC))*(1.0+A2)*FLOW(J)
6560=          GOTO 100
6570=     2    SUM = SUM + (.6*DIST(J)/GAMMA(JC))*FLOW(J)
6580=   100 CONTINUE
6590=       FPRIM = SUM
6600=       RETURN
6610=       END
6620=C
6630=C*************************************************************************
6640=C
6650=       REAL FUNCTION USERCST( J )
6660=C
6670=C--------CALCULATE USER COST = TRAVEL TIME FOR LINK J
6680=C
6690=       COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
6700=      +              NO(2840),ND(2840),NULACO(2840),DIST(2840)
6710=       COMMON /NET2/ VCUM(2840),TRAV(2840), FLOW(2840),ODIJ(150)
6720=       COMMON /COEF/ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
6730=C
6740=       VOL=VCUM(J)
6750=C--------UNPACK DELAY CURVE INDEX JC
6760=       JC=MOD(NULACO(J),1000)
6770=       IF (VOL.GE.0.0) GOTO 10
6780=       VOL = 0.0
6790=    10 CONTINUE
6800=       IF (GAMMA(JC).EQ.0.0) GOTO 2
6810=       IF (ALPHA(JC).EQ.0.0) GOTO 3
6820=       NULA=NULACO(J)/1000
6830=       DELT=DELTA(JC)*FLOAT(NULA)/10.0
6840=       IF (DELT.EQ.0.0) GOTO 2
6850=       USERCST=DIST(J)*.6/GAMMA(JC)*(1.0+ALPHA(JC)*(VOL/DELT)**BETA(JC))
6860=       RETURN
6870=     2 USERCST = 0.0
6880=       RETURN
6890=     3 USERCST = DIST(J)*.6/GAMMA(JC)
6900=       RETURN
6910=       END
6920=C
6930=C*************************************************************************
6940=C
6950=       SUBROUTINE LS(LAMBDA,DELTA,ACCUR,VAL,MODE)
6960=C
6970=C          FIND LAMBDA IN THE INTERVAL ZERO TO "DELTA" SUCH THAT
6980=C          THE DERIVATIVE OF THE OBJECTIVE FUNCTION IS ZERO.
6990=C          VAL IS SET TO THE CORRESPONDING OBJECTIVE FUNCTION
7000=C          VALUE.
```

```
7010=C
7020=       REAL LAMBDA
7030=C
7040=       XA = 0.0
7050=       XB = DELTA
7060=       FPXA = FPRIM(XA,MODE)
7070=       FPXB = FPRIM(XB,MODE)
7080=C
7090=       IF (FPXA*FPXB .LT. 0.0) GOTO 80
7100=          FXA = SUMTRA(XA,MODE)
7110=          FXB = SUMTRA(XB,MODE)
7120=          IF (FXA-FXB) 81,82,82
7130=    81       VAL = FXA
7140=             LAMBDA = XA
7150=             RETURN
7160=    82       VAL = FXB
7170=             LAMBDA = XB
7180=             RETURN
7190=C
7200=    80 CONTINUE
7210=C
7220=C-----BISECTION
7230=C
7240=       XMID = 0.5*(XA+XB)
7250=       FPXMID = FPRIM(XMID,MODE)
7260=       IF (FPXMID*FPXA .LT. 0.0) GOTO 83
7270=          XA = XMID
7280=          FPXA = FPXMID
7290=          GOTO 84
7300=    83    XB = XMID
7310=          FPXB = FPXMID
7320=    84 CONTINUE
7330=C
7340=C-----REGULA FALSI
7350=C
7360=       XSUIV = XB - ((XB-XA)/(FPXB-FPXA))*FPXB
7370=       FPXSUI = FPRIM(XSUIV,MODE)
7380=       IF (ABS(FPXSUI).GT.ACCUR) GOTO 85
7390=          FXSUIV = SUMTRA(XSUIV)
7400=          VAL = FXSUIV
7410=          LAMBDA = XSUIV
7420=          RETURN
7430=C
7440=    85 IF (FPXSUI*FPXA .LT. 0.0) GOTO 86
7450=          XA = XSUIV
7460=          FPXA = FPXSUI
7470=          GOTO 80
7480=    86    XB = XSUIV
7490=          FPXB = FPXSUI
7500=          GOTO 80
```

```
      END
C
C*****************************************************************
C
      SUBROUTINE NETWOR
C
C--------READ IN NETWORK DATA FOR ALL LINKS
C
      COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
     +              NO(2840),ND(2840),NULACO(2840),DIST(2840)
      COMMON /NET2/ VCUM(2840),TRAV(2840), FLOW(2840),ODIJ(150)
      COMMON        WKMEM(3300)
      COMMON /PARAM/ EPSI,TIMEMA,MAXCYC,MAXIT,IPCH,IPRT,IPOST,IFACS
      COMMON /STAT/ RUNSEC(10), TOTDEM
      COMMON /COEF/  ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
      COMMON /EDIT/ TITLE(12),RLINE(10),LINE(9),L1(26),L2(26)
      COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL,KEPFIL,MEMFIL
      COMMON /PARM2/ ACCUR,MAXPTS,NBITER,IPARTA
      INTEGER CARD,ODFIL,NETFIL,SOLFIL
      DIMENSION JDEST(150),JDEM(150)
      INTEGER DEFUNC
      EQUIVALENCE (NORIG,LINE(1)),(NDEST,LINE(2)),(IDIST,LINE(3)),
     +            (NULA,LINE(4)),(DEFUNC,LINE(5)),(ISPEED,LINE(6)),
     +            (ITOBS,LINE(7)),(IVOBS,LINE(8)),(LTYPE,LINE(9))
C
      INTEGER TITL1,TITL2,PARM,SIZE,FACS,PAR2
      DATA TITL1/"TIT1"/,TITL2/"TIT2"/,PARM/"PARM"/,SIZE/"SIZE"/,
     +     FACS/"FACS"/,PAR2/"PAR2"/
      DATA MAXERR/3/
C
 1000 FORMAT(10A8)
 1001 FORMAT(1X,A4,F10.5,F5.0,10I5)
 1002 FORMAT(1X,A4,3I5)
 1003 FORMAT(5X,3(I4,1X),I3,I2,1X,I4,1X,2(I5,1X),I4)
 1004 FORMAT(1X,A4,2F10.4,F10.1,F10.0)
 1005 FORMAT(1X,A4,2F10.4,F10.1,F10.0)
 1006 FORMAT(1X,A4,F10.5,3(I5))
 1010 FORMAT (1X,A4,  5A8/1X,A4,  7A8)
 2001 FORMAT(//,5X,"E03 LINKS NOT SORTED. LINE :",I4)
 2002 FORMAT(/5X,"E04 NO. CARDS .NE. NLINKS ",2I8)
 2051 FORMAT (5X,"E11 TITLE CARDS MISSING :  ",A5,5X,A5)
 2052 FORMAT (5X,"E12 PARAMETER CARD MISSING :  ",A5)
 2053 FORMAT (5X,"E13 SIZE CARD MISSING :  ",A5)
 2054 FORMAT(5X,"E14 DELAY FUNCTION FACTOR CARD MISSING :",I5)
 2055 FORMAT(5X,"E15 PARAMETER2 CARD MISSING :",I5)
 2060 FORMAT (5X,"E20 TOO MANY INPUT ERRORS :",I5)
 3001 FORMAT(2I5,10(I3,I4))
 3011 FORMAT(/,10X,"E05 ERROR IN INPUT OD MATRIX.  TYPE:",I3,
     +       "  CARD :",I5,"  ORIGIN :",I5)
 4001 FORMAT(/10X,"I07 PARTIAL SOLUTION INPUT",I3)
```

```
 4004 FORMAT(10F8.0)
C
      IERR=0
      ICARD=0
      READ 1010, ID, (TITLE(I), I=1, 5), ID2, (TITLE(I), I=6, 12)
      PRINT1010, ID, (TITLE(I), I=1, 5), ID2, (TITLE(I), I=6, 12)
      IF (ID.EQ.TITL1.AND.ID2.EQ.TITL2) GO TO 12
   10     PRINT 2051, ID, ID2
          IERR = IERR + 1
   12 READ 1000, RLINE
      PRINT1000, RLINE
      REWIND NETFIL
C
C--------READ ALGORITHM CONTROL PARAMETERS
C
      READ  1001,  ID, EPSI, TIMEMA, MAXCYC, MAXIT, IPOST, IPCH, IPRT, IFACS
      PRINT 1001,  ID, EPSI, TIMEMA, MAXCYC, MAXIT, IPOST, IPCH, IPRT, IFACS
      IF (ID .EQ. PARM) GO TO 22
   20     PRINT 2052, ID
          IERR = IERR + 1
   22 IF(EPSI.LE.0.0) EPSI=0.05
      IF(TIMEMA .LE. 0.0) TIMEMA =150.0
      IF(MAXCYC.LE.0) MAXCYC=20
      IF(MAXIT.LE.0) MAXIT=15
C
C--------READ DELAY FUNCTION FACTORS
C
      IF(IFACS.LE.0)GO TO 28
      DO 27 I=1, IFACS
      READ  1004, ID, ALPHA(I), BETA(I), GAMMA(I), DELTA(I)
      PRINT 1005, ID, ALPHA(I), BETA(I), GAMMA(I), DELTA(I)
      IF(ID.EQ.FACS)GO TO 27
   25     PRINT 2054, ID
          IERR=IERR+1
   27 CONTINUE
C
C--------READ NETWORK DATA AND CREATE NETWORK FILE
C
C
C LINE--1=NDA ORG, 2=NDA DEST, 3=LEN, 4=WIDTH, 5=DEFUNC, 6=SPEED, 7=TIME,
C       8=VOL, 9=TYPE, 10=TZERO, 11=CAP, 12=COO ORG, 13=COO DEST,
C       14=AOO ORG, 15=AOO DEST, 16=AOO LINK, 17=NDA LINK
C
   28 READ 1002,  ID, NNODE, NLINK, NCENT
      PRINT1002,  ID, NNODE, NLINK, NCENT
      IF (ID .EQ. SIZE) GO TO 32
   30     PRINT 2053, ID
          IERR = IERR + 1
   32 IF (IERR .LT. MAXERR) GO TO 36
   34     PRINT 2060, IERR
          CALL EXIT
```

```
8510=C
8520=      36 READ 1006, ID, ACCUR, MAXPTS, NBITER, IPARTA
8530=         PRINT 1006, ID, ACCUR, MAXPTS, NBITER, IPARTA
8540=         IF (ID.EQ.PAR2) GOTO 42
8550=            PRINT 2055, ID
8560=            IERR = IERR+1
8570=      42 IF (IERR.LT.MAXERR) GOTO 46
8580=            PRINT 2060, IERR
8590=            CALL EXIT
8600=C
8610=      46 DO 50 I=1,NNODE
8620=      50    NOLAB(I)=0
8630=         NOLAB(1)=1
8640=         I=0
8650=         J=1
8660=      55 I=I+1
8670=      54 READ(NETFIL,1003) LINE
8680=         IF(EOF(NETFIL).NE.0) GOTO 59
8690=C 1013 FORMAT(2016)
8700=C        WRITE(6,1013)LINE
8710=      56 ND(I) = NORIG
8720=         ND(I) = NDEST
8730=         DIST(I) = IDIST
8740=         NULACO(I) = NULA * 1000 + DEFUNC
8750=C--------SET UP LINK ORIGIN INDEX TABLE NOLAB
8760=         IF(ND(I)-J)57, 55, 58
8770=      57    PRINT 2001, I
8780=            CALL EXIT
8790=      58 J=ND(I)
8800=         NOLAB(J)=I
8810=         GOTO 55
8820=C
8830=      59 I=I-1
8840=         IF(I.EQ.NLINK)GO TO 61
8850=      60    PRINT 2002, I, NLINK
8860=            CALL EXIT
8870=C
8880=      61 CONTINUE
8890=         NOLD=0
8900=         TOTDEM = 0.0
8910=         DO 102 IO=1,NCENT
8920=C
8930=C-----READ IN THE DEMANDS ASSOCIATED WITH THE CURRENT ORIGIN
8940=C
8950=C        KD WILL CONTAIN ORIGIN NODE NUMBER OF EACH ORIGIN IN THE
8960=C           OD MATRIX IN THE ORDER READ IN.  KD ENTRY WILL BE NEGATIVE
8970=C           IF THERE IS NO DEMAND FROM THAT ORIGIN.
8980=C
8990=         DO 101 I=1,NCENT
9000=  101        ODIJ(I)=0.00
```

```
9010=        M=10
9020=        ICARD ICARD+1
9030=   103  READ (CARD,3001) N, NT, (L1(J), L2(J), J=1, 10)
9040=        IF(EOF(CARD). NE. 0) GOTO 105
9050=C        WRITE(6,3001)                    N, NT, (L1(J), L2(J), J=1, 10)
9060=        GO TO 106
9070=   105      IERR = 2
9080=            GO TO 113
9090=   106  IF(N. NE. 0. AND. N. NE. NOLD) GO TO 112
9100=        IF(N. EQ. 0)GO TO 103
9110=   111      IERR=1
9120=   113      PRINT 3011, IERR, ICARD, IO
9130=            CALL EXIT
9140=   112  KD(IO)=-N
9150=        NOLD=N
9160=C---------IF NO DEMAND FOR THIS ORIGIN KD ENTRY NEGATIVE
9170=        IF(NT. EQ. 0)GOTO 102
9180=        KD(IO)=N
9190=C---------SET UP DEMAND MATRIX ODIJ FOR THIS ORIGIN
9200=        DO 120 I=1, 10
9210=        JDEST(I)=L1(I)
9220=   120  JDEM(I)=L2(I)
9230=        IF(NT. LE. 10)GOTO 150
9240=        NUM=NT/10
9250=        IF(NUM*10. EQ. NT) NUM=NUM-1
9260=        DO 121 I=1, NUM
9270=        ICARD=ICARD+1
9280=        READ (CARD,3001) NX, NTX, (L1(J), L2(J), J=1, 10)
9290=        IF (EOF(CARD). NE. 0) GOTO 105
9300=C        WRITE(6,3001)                    NX, NTX, (L1(J), L2(J), J=1, 10)
9310=   126  IF(NX. NE. NOLD. AND. NX. NE. 0) GO TO 111
9320=        DO 122 J=1, 10
9330=        M=M+1
9340=        JDEST(M)=L1(J)
9350=   122  JDEM(M)=L2(J)
9360=   121  CONTINUE
9370=C
9380=   150   CONTINUE
9390=        DO 116 J=1, NT
9400=        IF(JDEST(J). EQ. N)GOTO 116
9410=        DEM=JDEM(J)
9420=        NEXT=JDEST(J)
9430=        ODIJ(NEXT)=DEM
9440=        TOTDEM = TOTDEM + DEM
9450=   116  CONTINUE
9460=        WRITE(ODFIL) ODIJ
9470=   102  CONTINUE
9480=C
9490=        RETURN
9500=C
```

```
9510=C
9520=C       RE-ESTARLISH PARTIAL SOLUTION
9530=C
9540=        ENTRY PARTSO
9550=        PRINT 4001, IPOST
9560=        REWIND SOLFIL
9570=        READ(SOLFIL, 1000) RLINE
9580=        PRINT 1000, RLINE
9590=        READ(SOLFIL, 4004) (VCUM(I), I=1, NLINK)
9600=        DO 206 I=1, NLINK
9610=   206  TRAV(I)=USERCST (I)
9620=        RETURN
9630=        END
9640=C
9650=C*********************************************************************
9660=C
9670=        SUBROUTINE LOADTR(NN)
9680=C
9690=C--------ACCUMULATE FLOW ON LINKSUSED ON ROUTE TO ALL DESTINATIONS
9700=C          FROM THE GIVEN ORIGIN (NN)
9710=C
9720=        COMMON /NFT1/ NNODE, NLINK, NCENT, KD(150), NOLAB(1100),
9730=       +              NO(2840), ND(2840), NULACO(2840), DIST(2840)
9740=        COMMON /NFT2/ VCUM(2840), TRAV(2840),  FLOW(2840), ODIJ(150)
9750=        COMMON        WKMEM(3300)
9760=        COMMON /UNIT/ CARD, ODFIL, NETFIL, SOLFIL, KEPFIL, MEMFIL
9770=        DIMENSION LPR(1100), TSUM(1100), NSET(1100)
9780=        EQUIVALENCE (TSUM(1), WKMEM(1)), (NSET(1), WKMEM(1101))
9790=        EQUIVALENCE (LPR(1), WKMEM(2201))
9800=        INTEGER WKMEM
9810=        INTEGER CARD, ODFIL, NETFIL, SOLFIL
9820=C
9830=C--------CONSTRUCT SHORTEST PATHS FROM ORIGIN NN TO ALL DESTINATIONS.
9840=C
9850=        CALL MINTRE (NN)
9860=C
9870=        LPR(NN)=0
9880=C
9890=C--------TRACE BACK LINKS ON SHORTEST PATH FOR EACH DESTINATION AND
9900=C          ADD OD VOLUME TO LINK TOTAL.
9910=C
9920=        DO 100 I=1, NCENT
9930=        IF(ODIJ(I).LE.0.0)GOTO 100
9940=        NEXT=I
9950=   104  L=LPR(NEXT)
9960=        IF(L.EQ.0)GOTO 100
9970=        FLOW(L)=FLOW(L)+ODIJ(I)
9980=        NEXT=NO(L)
9990=        IF(NEXT.NE.NN)GO TO 104
10000=  100  CONTINUE
```

```
9510=C
9520=C       RE-ESTARLISH PARTIAL SOLUTION
9530=C
9540=       ENTRY PARTSO
9550=       PRINT 4001, IPOST
9560=       REWIND SOLFIL
9570=       READ(SOLFIL,1000) RLINE
9580=       PRINT 1000,RLINE
9590=       READ(SOLFIL,4004) (VCUM(I), I=1,NLINK)
9600=       DO 206 I=1,NLINK
9610=   206 TRAV(I)=USERCST (I)
9620=       RETURN
9630=       END
9640=C
9650=C*****
9660=C
9670=       SUBROUTINE LOADTR(NN)
9680=C
9690=C--------ACCUMULATE FLOW ON LINKSUSED ON ROUTE TO ALL DESTINATIONS
9700=C          FROM THE GIVEN ORIGIN (NN)
9710=C
9720=       COMMON /NFT1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
9730=      +              NO(2840),ND(2840),NULACO(2840),DIST(2840)
9740=       COMMON /NFT2/ VCUM(2840),TRAV(2840), FLOW(2840),ODIJ(150)
9750=       COMMON        WKMEM(3300)
9760=       COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL,KEPFIL,MEMFIL
9770=       DIMENSION LPR(1100),TSUM(1100),NSET(1100)
9780=       EQUIVALENCE (TSUM(1),WKMEM(1)),(NSET(1),WKMEM(1101))
9790=       EQUIVALENCE (LPR(1),WKMEM(2201))
9800=       INTEGER WKMEM
9810=       INTEGER CARD,ODFIL,NETFIL,SOLFIL
9820=C
9830=C--------CONSTRUCT SHORTEST PATHS FROM ORIGIN NN TO ALL DESTINATIONS.
9840=C
9850=       CALL MINTRE (NN)
9860=C
9870=       LPR(NN)=0
9880=C
9890=C--------TRACE BACK LINKS ON SHORTEST PATH FOR EACH DESTINATION AND
9900=C          ADD OD VOLUME TO LINK TOTAL.
9910=C
9920=       DO 100 I=1,NCENT
9930=       IF(ODIJ(I).LE.0.0)GOTO 100
9940=       NEXT=I
9950=   104 L=LPR(NEXT)
9960=       IF(L.EQ.0)GOTO 100
9970=       FLOW(L)=FLOW(L)+ODIJ(I)
9980=       NEXT=NO(L)
9990=       IF(NEXT.NE.NN)GO TO 104
10000=  100 CONTINUE
```

```
10010=C
10020=       RETURN
10030=       END
10040=C
10050=C*************************************************************************
10060=C
10070=       SUBROUTINE SUMS(WEWSUM,GRADX,GRADY)
10080=C
10090=       COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
10100=      +              NO(2840),ND(2840),NULACO(2840),DIST(2840)
10110=       COMMON /NET2/ VCUM(2840),TRAV(2840),  FLOW(2840),ODIJ(150)
10120=C
10130=       GRADX = 0.0
10140=       GRADY = 0.0
10150=       DO 10 J=1,NLINK
10160=          GRADX = GRADX + VCUM(J)*TRAV(J)
10170=          GRADY = GRADY + FLOW(J)*TRAV(J)
10180=          X = FLOW(J) - VCUM(J)
10190=          FLOW(J) = X
10200=    10 CONTINUE
10210=       WEWSUM = GRADY - GRADX
10220=       RETURN
10230=       END
10240=C
10250=C*************************************************************************
10260=C
10270=       SUBROUTINE EXFLOW
10280=C
10290=       COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
10300=      +              NO(2840),ND(2840),NULACO(2840),DIST(2840)
10310=       COMMON /NET2/ VCUM(2840),TRAV(2840),  FLOW(2840),ODIJ(150)
10320=       COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL,KEPFIL,MEMFIL
10330=       INTEGER CARD,ODFIL,NETFIL,SOLFIL
10340=C
10350=       REWIND ODFIL
10360=C
10370=       DO 190 I=1,NLINK
10380=   190 FLOW(I)=0.0
10390=C
10400=C
10410=       DO 42 N=1,NCENT
10420=          NN=KD(N)
10430=          IF(NN.LE.0)GOTO 42
10440=C--------READ DEMAND FOR THIS ORIGIN
10450=          READ (ODFIL) ODIJ
10460=C--------ACCUMULATE ALL OR NOTHING SOLUTION (FLOW) FOR THIS ORIGIN
10470=          CALL LOADTR(NN)
10480=    42    CONTINUE
10490=       RETURN
10500=       END
```

```
10510=C
10520=C*****************************************************************
10530=C
10540=       SUBROUTINE UPDTGT (LAMBDA,LIMKEP )
10550=C
10560=       COMMON/OMEGAS/OMEGA(31),DN(31),H(31)
10570=       REAL LAMBDA
10580=C
10590=       DO 80 I=1,LIMKEP
10600=          OMEGA(I)=OMEGA(I)+LAMBDA*DN(I)
10610=          IF (I.EQ 1) GOTO 10
10620=          PRINT 999, I, OMEGA(I)
10630=          GOTO 80
10640=    10    PRINT 998, I, OMEGA(I)
10650=    80 CONTINUE
10660=   998 FORMAT(/28X,"OMEGA(",I2,") = ",F12.9)
10670=   999 FORMAT(28X,"OMEGA(",I2,") = ",F12.9)
10680=C
10690=       RETURN
10700=       END
10710=C
10720=C*****************************************************************
10730=C
10740=       SUBROUTINE UPDATQ(LIMKEP ,LAMBDA)
10750=C
10760=       COMMON/OMEGAS/OMEGA(31),DN(31),H(31)
10770=       REAL LAMBDA
10780=C
10790=       X=1.0 - LAMBDA
10800=       PRINT 99,LAMBDA
10810=    99 FORMAT(/21X,"APPEL DE UPDATQ  ===,LAMBDA :  ",F14.9)
10820=       NM1=LIMKEP -1
10830=       DO 10 I=1,NM1
10840=          OMEGA(I)=OMEGA(I)*X
10850=          PRINT 999, I, OMEGA(I)
10860=    10 CONTINUE
10870=   999 FORMAT(28X,"OMEGA(",I2,") = ",F12.9)
10880=       OMEGA(LIMKEP )=LAMBDA
10890=       PRINT 999,LIMKEP,OMEGA(LIMKEP)
10900=       RETURN
10910=       END
10920=C
10930=C*****************************************************************
10940=C
10950=        SUBROUTINE NEWFLO (NEWSUM,LIMKEP )
10960=C
10970=       COMMON/OMEGAS/OMEGA(31),DN(31),H(31)
10980=       COMMON /NET1/ NNODE,NLINK,NCENT,KD(150),NOLAB(1100),
10990=      +              NO(2840),ND(2840),NULACO(2840),DIST(2840)
11000=       COMMON /NET2/ VCUM(2840),TRAV(2840), FLOW(2840),ODIJ(150)
```

```
11010=       COMMON          WKMEM(3300)
11020=       COMMON /UNIT/ CARD, ODFIL, NETFIL, SOLFIL, KEPFIL, MEMFIL
11030=       INTEGER CARD, ODFIL, NETFIL, SOLFIL
11040=
11050=       JMAX=0
11060=       ALMAX=0.
11070=       DO 15 I=1, LIMKEP
11080=    15 H(I)=0.
11090=       REWIND KEPFIL
11100=C
11110=       DO 20 I=1, LIMKEP
11120=       READ(KEPFIL)FLOW
11130=       DO 25  J=1, NLINK
11140=    25 H(I)=H(I)+TRAV(J)*FLOW(J)
11150=       IF(ALMAX. GE. OMEGA(I))GOTO 20
11160=       ALMAX=OMEGA(I)
11170=       JMAX=I
11180=    20 CONTINUE
11190=C
11200=    30 SUM=0. 0
11210=       SIGMA= - 1. 0E10
11220=C
11230=       DO 40 I=1, LIMKEP
11240=       IF(I. EQ. JMAX)GOTO 40
11250=       DN(I)=H(JMAX)-H(I)
11260=       HQ=DN(I)
11270=       IF(HQ. GE. 0. )GOTO 35
11280=       DN(I)=OMEGA(I)*HQ
11290=       IF(OMEGA(I). LE. 0. 001)DN(I)=0.
11300=    35 SUM=SUM-DN(I)
11310=       IF(DN(I). GE. 0. 0)GOTO 40
11320=       X= OMEGA(I)/DN(I)
11330=       IF(SIGMA. LT. X)SIGMA= X
11340=    40 CONTINUE
11350=C
11360=       DN(JMAX)=SUM
11370=       IF (SUM. LT. 0. 0. AND. ALMAX/SUM. GT. SIGMA)SIGMA=ALMAX/SUM
11380=       SIGMA= - SIGMA
11390=       DO 50 I=1, LIMKEP
11400=    50 DN(I)=DN(I)*SIGMA
11410=C
11420=       DO 55 I=1, NLINK
11430=    55 FLOW(I)=0. 0
11440=       REWIND KEPFIL
11450=C
11460=       DO 70 J=1, LIMKEP
11470=       READ(KEPFIL)TRAV
11480=        DO 60 I=1, NLINK
11490=    60  FLOW(I)=FLOW(I)+DN(J)*TRAV(I)
11500=    70 CONTINUE
```

```
11510=C
11520=       NEWSUM=0.
11530=       DO 90 I=1,NLINK
11540=       TRAV(I)=USERCST (I)
11550=       NEWSUM=NEWSUM+TRAV(I)*FLOW(I)
11560=    90 CONTINUE
11570=       RETURN
11580=       END
```

20.42.43. T361950. 3619-A7 EG(02-PTX) PR 1 1160 LN 19 P? 83.05/03 MFB/ A7 S1093